DISTRIBUTED LOAD BALANCING IN MANY-TO-ONE WIRELESS SENSOR NETWORKS

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Engineering and Physical Sciences

2013

By
Anthony Kleerekoper
School of Computer Science

Contents

Word Count: 46271

List of Tables

List of Figures

Abstract

A typical sensor network is conceived as being a very large collection of low-powered, homogeneous nodes that remain static post-deployment and forward sensed data to a single sink via multi-hop communication. For these types of networks there is an inherent funnelling effect whereby the nodes that can communicate directly with the sink must collectively forward the traffic of the entire network and therefore these nodes use more energy than the other nodes. This is known as the energy hole problem because after some time, these nodes deplete their batteries and leave an energy hole cutting the sink off from the network.

In this thesis two new routing protocols are proposed that aim to maximise load balancing among these most critical nodes in order to maximise lifetime. They are the first fully distributed routing protocols that are designed to generate a load balanced routing tree to mitigate the energy hole problem. The results show that the better performing of the two is capable of creating a highly balanced tree at the cost of a small increase in latency.

Although there have been other fully distributed protocols that aim at a similar form of load balancing, it is proven that the approach they take cannot guarantee perfect balance among the most critical nodes even in unrealistically generous scenarios. This suggests that they are not well suited to that task and the simulation results show that the novel protocols proposed in this thesis outperform the best of the alternatives.

Before these protocols are proposed, the absolute reception-based blacklisting routing strategy is shown to be more energy efficient than previously thought and indeed more efficient than the strategy that has previously been considered optimal. This result is used to strongly justify the use of the unit disk graph

model in simulations of sensor networks. Additionally, the relay hole problem in sensor networks is analysed for the first time.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

I would like to thank my supervisor, Dr Nick Filer, for his specific help with the work in this thesis and also for his guidance and mentoring. Thanks also to Dr Barry Cheetham for his invaluable insights and suggestions throughout my PhD studies and to my colleagues and friends at the university for providing sound boards and light relief. Special thanks to Marci Freedman who took time out of her own research project to proof-read this thesis. I am also grateful to the EPSRC for providing financial support throughout my studies.

Words cannot express the gratitude I have to my parents for the way they raised me; without them I simply would not be where I am now. Likewise, untold thanks are due to my dear wife Naomi who offered words of encouragement when they were needed and silence when that was. This thesis would not have been written without her help and support. Thanks also to my son Avi who provided plenty of alternative entertainment and joy when I needed to take a break from research (and when I didn't too)!

Above all else, though, I wish to express my gratitude to G-d who controls all and provides all. I am thankful that He has guided me on my path in life so far and pray that He will continue to shower me and my family with all that we need. May my desires always be His desires.

Chapter 1

Introduction

According to Romer and Mattern, early research into wireless sensor networks (WSN or *sensor networks*) resulted in the following de facto definition of a WSN as a:

> "large-scale (thousands of nodes, covering large geographical areas), wireless, ad hoc, multihop, unpartitioned network of homogeneous, tiny (hardly noticeable), mostly immobile (after deployment) sensor nodes that would be randomly deployed in the area of interest." [RM04]

However, as Sadler pointed out, "given any definition of a sensor network, there exists a counter example."[Sad05] and Martin and Paterson have simply concluded that "there is no single, precise, definition of a wireless sensor network." [MP08]

Nevertheless, Buratti *et al.* have given a general definition of a sensor network as:

> "a network of devices, denoted as *nodes*, which can sense the environment and communicate the information gathered from the monitored field (e.g., an area or volume) through wireless links. The data is forwarded, possibly via multiple hops, to a sink (sometimes denoted as controller or monitor) that can use it locally or is connected to other networks (e.g., the Internet) through a gateway. The nodes can be stationary or moving. They can be aware of their location or not.

They can be homogeneous or not." [BCDV09]

The WSN market is expected to experience massive growth in the coming years. A recent market research report claims that wireless mesh networks will undergo a compound annual growth rate (CAGR) of 16.1% to reach $2bn by 2021 [IDT]. Another report argues that the Industrial WSN market will be worth $3.795bn by 2017, experiencing a CAGR of 15.58% [Mar] whilst a third report on wireless sensor devices predicts a 43.1% CAGR leading to a market worth $4.7bn by 2016 [Res].

Sensor networks offer numerous advantages over more traditional sensing solutions, particularly for data gathering applications. These include the ability to deploy a larger number of nodes for the same price which allows for sensor coverage of a wider area. Individual sensors can be closer to the phenomenon being investigated by virtue of having a higher density yet the devices are less obtrusive so that they have less of an impact on the environment they are measuring. Sensor networks may also be made to be self-organising and autonomous with high fault tolerance which makes them easier to deploy and extend and allows them to be used in harsh or hostile environments.

Among the first examples of sensor networks were the Great Duck Island experiments [MCP+02], sniper detection [SML+04] and zebra monitoring [JOW+02] systems. More recent examples are networks such as the SFPark program in San Fransisco [SFP] and the Siega System agricultural management system [Sie].

One of the main challenges for these networks is energy management because in many cases the sensor devices are battery operated and the batteries cannot be replaced. This could be because the network consists of so many nodes that replacing all depleted batteries is not feasible or because the network is located in a remote or hostile environment. Although some sensor networks can be mains powered, for many data gathering applications the networks will be deployed into areas without the required infrastructure which means that energy must be provided either by batteries or through some form of energy harvesting eg solar cells. However, even in situations where energy harvesting is possible, energy usage must still be carefully managed as the available energy remains limited.

In order to maximise their lifetime, individual nodes must use their energy resources carefully while still completing their set tasks. However, even if the energy

consumption of each individual node is minimised as far as possible, there are still important steps that need to be taken to increase network lifetime. Foremost among these is to balance the work load among the nodes of the network to prevent some nodes prematurely running out of energy.

In this thesis I focus on the question of load balancing; in particular, load balancing in many-to-one WSNs that use multi-hop communication, such as might be expected for monitoring applications. Examples of this type of application include volcano monitoring [HSX+12], greenhouse monitoring [AVE08] and GlacsWeb [Gla]. These applications can often involve many hundreds of identical nodes deployed over a large area designed to collect data samples periodically. The ongoing VolcanoSRI project is an example of the kind of networks being considered [Vol]. This project plans to deploy a 500 node network to monitor seismic activity on a volcano in Ecuador. All the nodes will be identical and deployed in a roughly uniform, circular network as illustrated in Fig. 1.1. In this example the nodes communicate with Bluetooth and will be powered by four D-Cell batteries.

Other specific examples include a planned 300 node agricultural monitoring network [WWQ+10]. Again the nodes are all intended to be identical and use low-powered radios. In this case an RF230 radio is intended which would provide a maximum transmission range of 300m. Rather than batteries, the nodes are solar powered which allows for much longer lifetime but places a strict limit on peak energy use.

With sensor networks it is possible to define four types of load balancing in reference to the corona model, as illustrated in Fig. 1.2. This model, which will be more fully described in Chapter 3, provides a method for mathematically analysing a sensor network. The sink is assumed to be at the centre of the network surrounded by the sensor nodes which all share the same transmission range. As a result, the nodes that are within that range of the sink can communicate directly with it but all other nodes must use relays. This gives rise to a series of concentric coronas of the same width as the transmission range. Nodes in a given corona use other nodes in the next inward corona as relays for multi-hop communication to the sink. Conveniently, a node that is in corona x is also x hops away from the sink.

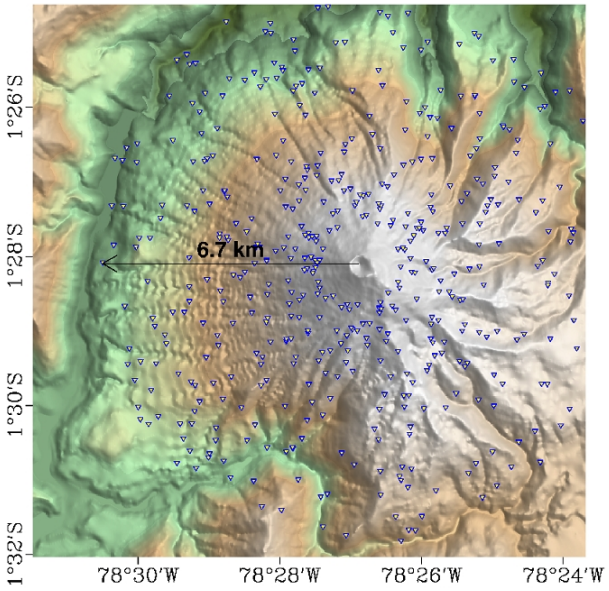


Figure 1.1: The ongoing VolcanoSRI project aims to deploy a 500 node network to measure seismic activity on a volcano in Ecuador. This project is of the kind that are being considered in this thesis.

Zhang and Shen use the corona model to label two main types of load balancing namely *inter-corona* and *intra-corona* balance [ZS09]. A third type of load balancing appears, unnamed, in the literature and I refer to it in this thesis as *degree balance*. Finally, the focus of this thesis is on a variant of intra-corona balance that I label *inner-corona* balance which is the fourth type of load balancing.

Inter-corona balance is the optimal type and is achieved when all nodes in all coronas perform the same amount of work since all the nodes will deplete their batteries at the same time leaving no residual energy left unused in the network. However, as will be discussed more fully in the next chapter, inter-corona balance is not always possible. In particular, for sensor networks that accord with the de facto definition quoted above from Romer and Mattern inter-corona imbalance is inevitable [SNK05].

Intra-corona balance is a component of inter-corona balance but can exist independently. For intra-corona balance to be achieved all the nodes within the same corona must perform the same amount of work, even if this is different to the amount performed by nodes in other coronas. Intra-corona balance has typically been studied as a component for inter-corona balance rather than as an independent goal.

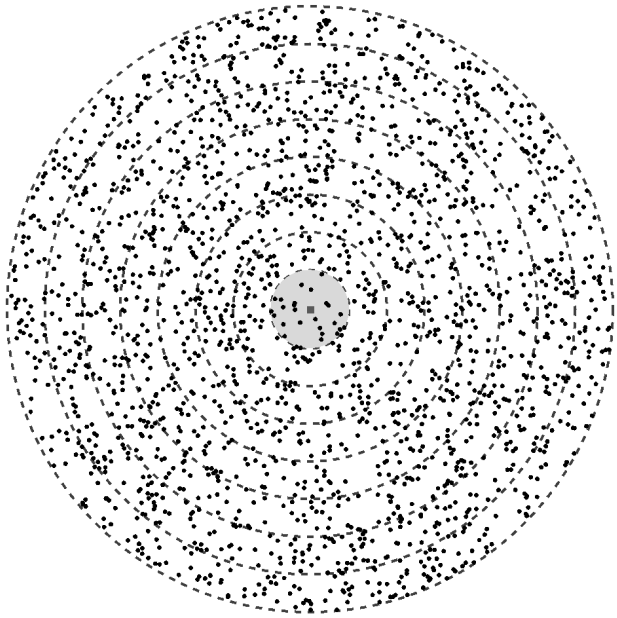


Figure 1.2: A circular sensor network can be viewed as a series of concentric coronas. The square in the centre is the sink. The shaded corona contains the most critical nodes that will deplete their batteries first, cutting off the sink from the rest of the network.

Degree balance can be viewed as a kind of intra-corona balance in that its focus is on reducing variation between nodes within the same corona rather than between coronas. However, while intra-corona balance aims to reduce variation in work rates, degree balance deals only with the node degree. Degree balance assumes that the network traffic is many-to-one and that a static routing tree is being used so that all nodes have only a single parent. With these assumptions, a node's degree is a measure of the number of children it has in the routing tree which can serve as a proxy for its work rate. In a data gathering network where every node generates the same traffic, the difference between the two types is that degree balance reduces variation in the number of children per node and intra-corona balance reduces variation in the number of descendants per node. Degree balancing uses local information whereas intra-corona balance uses more global knowledge.

The final type of load balancing, inner-corona balance, is a sub-problem of intra-corona balance, that is concerned only with the inner-most corona of the network. The aim is to minimise variation in the workload among the nodes in the inner-most corona without directly being concerned with the balance of other parts of

the network.

1.1 Aims and Motivation

The aim of this thesis is to investigate a new approach for lifetime maximisation in sensor networks. This approach involves proposing novel, distributed protocols which create a static routing tree that maximises inner-corona balance. The work is motivated by the absence of any such protocols in the literature despite the advantages that they appear to offer. While numerous protocols have been proposed that are distributed, produce static routing trees or that maximise inner-corona balance; to the best of my knowledge the protocols in this thesis are the first to combine all three properties.

The theoretical advantages of each of the properties will be more fully discussed in Chapter 2 but are briefly described here. Distributed protocols utilise only local information which reduces the initial communication costs when compared to centralised solutions. For a centralised solution, the sink would need to have accurate topological information from the entire network meaning that every node must inform the sink about the nodes it is able to communicate with. Although the amount of data involved is significantly less than the total amount of data that is expected to be collected by the network, it is still a larger cost than is incurred by a distributed solution.

The problem of collecting the initial information is hampered further by the lack of any existing routing tree. In its place some form of flooding would be required to guarantee that the data reaches the sink which increases the overheads still further. As the network becomes more dense this disparity increases as the amount of data being sent by each node increases as does the impact of flooding.

Nevertheless, all these costs remain one-off initialisation costs which are relatively insignificant to the total amount of communication. If a centralised solution is capable of providing a significantly better solution to the problem then the costs may be a small price to pay. There is a trade-off between the costs of gathering the information and the quality of the solution. Centralised solutions have been proposed in the past but this thesis focuses on distributed solutions which may

be capable of providing strong solutions with reduced overhead.

Static routing trees are also a means of reducing energy consumption through reduced communication overhead. A static tree is created once and used for an extended period, ideally until the nodes deplete their batteries. In dynamic routing schemes, all nodes must maintain a routing table with up-to-date information about its potential parents in order to make sensible decisions. Keeping the table's contents fresh requires the regular sharing of information among neighbours which is the communication overhead. Nodes in a static routing tree have only a single parent to use for the duration of the tree's lifetime and therefore do not need to be updated with information from neighbours.

The final property is that the protocols aim to maximise inner-corona balance and this has advantages over the other types of balance. The characteristics of the networks studied in this thesis are detailed in Chapter 3 and for these types of networks inter-corona balance is impossible [SNK05]. In brief, these networks consist of homogeneous, static nodes that all generate data at the same rate. The generated data is transmitted through multiple hops to a single sink without using perfect aggregation (that is, the number or size of packets transmitted by a node is larger than the number or size of all packets received because locally generated data must be added). These characteristics make it impossible to achieve inter-corona balance as will be discussed more fully in Chapter 2.

It is simple to prove that for these networks the node that will deplete its batteries first is always in the inner-most corona. Let us do so by considering an arbitrary node A which is the node in the network that performs the most communication work per time unit. If node A is not in the inner-most corona then it must have a parent node, B, to which it forwards all its data. But since all nodes output more data than they receive, this node B would be receiving more data than node A does and transmitting more data as well which contradicts the original definition of node A. It must be, therefore, that node A is in the inner-most corona. Since all nodes start with the same initial energy and the energy consumption from communication is the dominant energy use in sensor devices, the node that performs the most communication work per time unit will deplete its batteries first and this node is always in the inner-most corona.

The significance of this observation is that intra-corona balance provides no greater lifetime than inner-corona balance since lifetime is typically measured

as the time until the first node depletes its batteries (dies). However, intra-corona balance requires some global knowledge because complete work-loads must be known in order to be balanced and this prevents intra-corona balance being achieved with fully distributed protocols. Since inner-corona balance can be approached with a distributed protocol and achieves the same network lifetime, it is obviously advantageous to focus on this type of balance over intra-corona balance.

The final alternative to consider is degree balancing which can be maximised using only local knowledge. The problem with degree balancing is that a node's total work depends not only on its degree but on the total number of descendants it has in the routing tree. As a result, a routing tree with perfect degree balance cannot guarantee to provide perfect inner-corona balance and may therefore have a sub-optimal lifetime. On the other hand, a routing tree with perfect inner-corona balance guarantees maximum network lifetime. Although a distributed algorithm is unlikely to produce perfect balance of any type owing to its imperfect information, it seems likely that an approach than cannot theoretically offer maximum lifetime will result in shorter practical lifetime than an approach that can theoretically offer maximum lifetime.

Thus the motivation for this thesis is the hypothesis that the lifetime of some types of sensor networks are longest when routing is through a static routing tree with maximum inner-corona balance created by a distributed algorithm.

1.2 Research Contributions

The main contribution of this thesis is the proposal and analysis of a novel distributed routing protocol, DECOR (for DEgree COnstrained Routing), which constructs a static routing tree designed to maximise inner-corona balance. This and a number of other contributions are briefly outlined in this section in the order they appear in the thesis.

1: Revisiting Blacklisting for Energy-Efficient Position Based Routing

Position based routing is a common paradigm for routing in wireless sensor networks. Nodes are assumed to know their location, either through GPS or other localisation techniques, sharing this information with their one hop neighbours. Each node can select its parent based on the amount of progress made towards the sink. This method of routing can reduce the amount of overhead required and results in scalable routing protocols.

A widely used model for wireless communication, known as the unit disk graph (UDG) model, states that two nodes can communicate perfectly if the distance between them is below some specified threshold but if the distance is above the threshold then no communication is possible [BFN01, KWZ03]. The quality of communication between nodes can be measured by the packet reception rate (PRR) which is the ratio of packets transmitted that are received. The UDG model includes two regions around a transmitting node: the connected region which extends up to the threshold distance and the disconnected region outside that distance. The PRR in the connected region is always 100% and it is always 0% in the disconnected region.

However, it was noted that in reality a third region exists in between these two called the transitional region [ZK04]. The PRR in the transitional region varies widely and although the average PRR falls predictably with the distance of the receiver from the transmitter, the actual PRR of a given receiver is hard to predict from distance. Fig. 1.3 shows the way in which PRR varies with distance and illustrates the three regions.

The recognition of the transitional region opened up the question of whether progress was the only factor that should be considered during position based forwarding. A trade-off was noted between progress and energy efficiency. If links were chosen which made the most progress then that was likely to result in selecting parents from inside the transitional region where errors might be frequent resulting in retransmissions and wasted energy. On the other hand, by selecting more reliable links that were closer to the transmitting node, more hops would be required. In the end, research seemed to indicate that the most energy efficient method was to consider both the progress made by a link and its packet

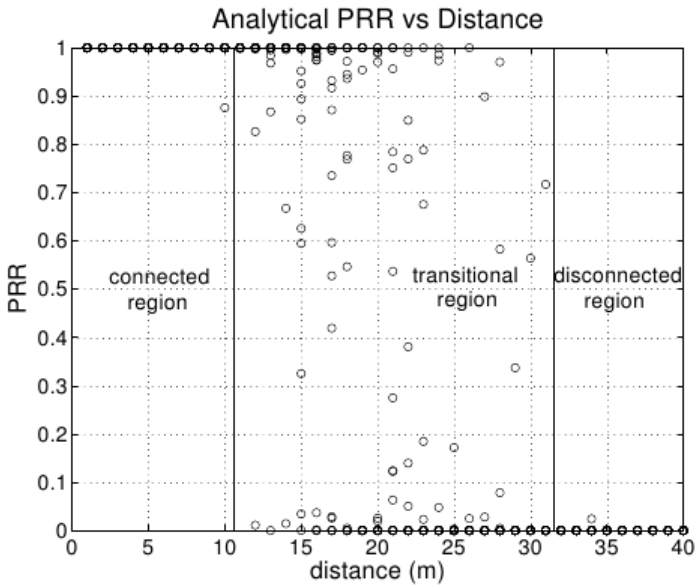


Figure 1.3: In the real world, three distinct regions exist around a transmitting node each displaying different behaviours of the packet reception rate (PRR). Image taken from [ZK04].

reception rate (PRR). One metric that was suggested, for example, was to select the link with the largest PRR×progress value [SZHK04].

That conclusion is revisited, based on the observation that automatic repeat request (ARQ) should not be considered as a network wide decision (as the previous researchers did) but as a function of the link quality. That is, ARQ need only be used if the link quality is unacceptably low and should not be used on high quality links where no benefit is gained. This implies that previous studies may have underestimated the additional cost of using low-PRR links and that a different strategy would therefore be more efficient.

Contribution 1 of this thesis is to show that the most energy-efficient method of selecting links for position-based routing is to use absolute reception-based blacklisting (ARB) to exclude low quality links and then select the link which makes the most progress from the non-blacklisted links.

2: Justifying the Unit Disk Graph Model

The UDG model includes only two of the three regions surrounding a transmitting node and was widely shown to be inaccurate some time ago [GKW+02, ZG03, WTC03, ZHKS04, CABM05]. Despite this, the model remains widely used because of its simplicity and usefulness in mathematical analysis. There would appear to be a need to justify its use in light of its inaccuracy and to demonstrate that, with certain caveats given below, results obtained using UDG are reliable.

One simple approach to justifying its use is to argue that that UDG model is correct up to the connected region and, therefore, results derived from it are reliable if the transmission range of nodes is limited to the connected region. However, this would limit all such results to sub-optimality since there are almost always some longer links available with high PRR.

Based on contribution 1 showing that ARB is more energy-efficient than alternative schemes for position-based routing, it is possible to provide a stronger justification for the unit disk graph model as an approximation of this forwarding strategy. Contribution 2 of this thesis is to show that the UDG model is a close approximation to the performance of ARB and that results derived using it are reliable.

3: The Relay Hole Problem

One of the central assumptions of the corona model, described above in Section 1 and more fully in Chapter 2, is that every node in one corona uses a node in the next inward corona to forward its packets towards the sink [OS06]. This assumption is justified on the basis that the node density in sensor networks is so high that large gaps cannot exist in the network. However, even for very large densities, some gaps will still exist and they can still result in nodes being unable to forward their packets into the next corona. I refer to this as the *relay hole problem* [KF12a].

Contribution 3, is to analyse the relay hole problem in sensor networks and show that while large densities mitigate it, they do not remove the problem completely. The effect of the relay hole problem is to increase the latency of the network by

increasing the average hop count.

4: Degree Balance and Inner-Corona Balance

A number of proposed routing protocols aim to maximise degree balance, that is to minimise the variation in the number of children adopted by nodes in the same corona [APZY+09, HCWC09, CZYG10]. However, none of them directly considers the question of inner-corona balance and therefore it remains unclear whether maximising degree balance is an efficient approach to maximising inner-corona balance.

Contribution 4, is to prove that the degree balancing approach cannot guarantee perfect inner-corona balance even when idealistic assumptions are made about the network. This is important because it is likely that an approach that cannot guarantee perfect balance under any circumstances will result in lower balance than an approach that can make this guarantee.

5: Role Based Routing

Contribution 5 of this thesis is to propose the first of two novel, distributed routing protocols designed to maximise inner-corona balance. The method, ROBAR (ROle BAsed Routing), works by assigning quotas to nodes specifying how many children they may adopt. Different nodes are assigned different quotas which define their role within the network.

The approach is proved to provide perfect inner-corona balance in idealised circumstances which suggests that it should perform better than the protocols that aim to maximise degree balance only. This too is shown but the cost of the increased balance is that not all nodes in the network are able to connect to the routing tree.

6: Degree Constrained Routing

Contribution 6 is the main contribution of this thesis, namely the novel routing protocol DECOR (DEgree COnstrained Routing). Along similar lines to ROBAR, DECOR increases balance by assigning quotas to nodes but the quotas are

assigned based on the node's level in the routing tree.

The approach is proved to provide perfect inner-corona balance in idealised circumstances but at the cost of connectivity or latency. In more realistic scenarios DECOR still performs better than alternative protocols. Additional techniques are proposed that result in a version of DECOR that provides full connectivity and high balance in exchange for a modest increase in latency. This protocol is analysed through extensive simulations in numerous scenarios.

1.3 Published Papers

The following peer-reviewed papers have been published based on the research in this thesis:

1. [KF12b] Kleerekoper, A.; Filer, N.; ,"Revisiting Blacklisting and Justifying the Unit Disk Graph Model for Energy-Efficient Position-Based Routing in Wireless Sensor Networks," Wireless Days (WD), 2012 IFIP , vol., no., pp.1-3, 21-23 Nov. 2012

   This paper forms part of Chapter 3.

2. [KF12a] Kleerekoper, A.; Filer, N.; , "The Relay Area Problem in Wireless Sensor Networks," Computer Communications and Networks (ICCCN), 2012 21st International Conference on , vol., no., pp.1-5, July 30 2012-Aug. 2 2012

   This paper forms part of Chapter 4.

3. [KF12c] Kleerekoper, A.; Filer, N.; ,"Trading latency for load balancing in many-to-one wireless networks," Wireless Telecommunications Symposium (WTS), 2012 , vol., no., pp.1-9, 18-20 April 2012

   This paper forms part of Chapter 7.

1.4 Thesis Structure

- Chapter 2 goes through the existing literature giving a summary of the well researched inter-corona balance problem and a complete treatment of the

much sparser research into the other forms of balance.

- Chapter 3 lays out the system model and assumptions that are used throughout the thesis. Included in this chapter are the first two contributions.
- Chapter 4 describes the third contribution regarding the relay hole problem.
- Chapter 5 proves the fourth contribution regarding degree balance.
- Chapter 6 proves the fifth contribution regarding role based routing.
- Chapter 7 proves the sixth contribution regarding degree constrained routing.
- Chapter 8 extends the analysis of DECOR by moving beyond the corona model.
- Chapter 9 summarises the contributions and outlines future avenues for research.

Chapter 2

Literature Review

As discussed in the previous chapter it is possible to define four types of load balancing in data gathering sensor networks with reference to the corona model. The ideal is inter-corona balance in which all nodes in all parts of the network perform approximately the same amount of work and deplete their batteries at the same rate as this makes full use of all the network's energy resources. However, in data gathering networks where data flows from the nodes to a single sink without perfect aggregation there is an inherent load imbalance which causes the nodes closest to the sink to deplete their batteries earlier than the other nodes. This is known as the energy hole problem.

In a network affected by the energy hole problem it is impossible to achieve inter-corona balance because of the inherent load imbalance [SO05] which leaves the remaining three types of balance: intra-corona, degree and inner-corona. In terms of network lifetime there is no advantage to intra-corona or degree balancing over inner-corona balance as proved earlier in Section 1.1. Therefore, the primary aim of this thesis is to propose novel, distributed routing protocols that can achieve improved levels of inner-corona balance compared to existing protocols.

This chapter reviews the existing literature concerning load balancing in sensor networks with two aims in mind. Firstly, it will show that inter-corona balance is not possible in all networks by reviewing the proposed solutions to the energy hole problem and highlighting the network conditions that must exist for each solution to be viable. Secondly, the need for new routing protocols will be demonstrated by noting the lack of protocols that are fully distributed, static and maximise

inner-corona balance.

In the next section the corona model and the assumptions underpinning it are described in more detail and the model is used to analyse the load balancing problems in sensor networks. In Section 2.2 the conditions needed to provide inter-corona balance are highlighted by briefly reviewing a sample of solutions to the energy hole problem. Section 2.3 discusses dynamic routing which can provide all three other forms of load balancing but at the cost of increased overhead. Section 2.4 gives a thorough review of the proposed solutions to the degree balance problem. Section 2.5 describes the centralised and semi-distributed algorithms that can provide inner-corona balance.

2.1 The Corona Model and the Energy Hole Problem

The first use of concentric circles, or coronas, with regards to sensor networks appears to be by Wadaa *et al.* [WOW+03] who proposed a training scheme to divide a network into clusters without the use of location information or node IDs. The method assumes that the sink can communicate with all nodes but that the nodes must use multi-hop communication. The sink also has a number of different transmission power levels to choose from and can narrow its antenna to make it highly directional. With these abilities the sink can divide the network into clusters by first dividing it into coronas and then wedges, as illustrated in Fig. 2.1.

In order to split the network into coronas the sink repeatedly broadcasts beacons at ever increasing transmission power levels. The nodes that receive the first beacon, sent at the lowest level, are in the first corona; those that receive only the second beacon are in the second corona and so on. To create the wedges, the sink directs its antenna to one portion of the network and transmits a beacon at its maximum transmission power level. This beacon contains a wedge identifier so that nodes receiving it can identify which wedge they are in. The sink then changes the angle of directionality of the transmission and rebroadcasts with a new wedge identifier and this continues until the sink has broadcast to the entire network. When the process completes, every node has received at least one beacon

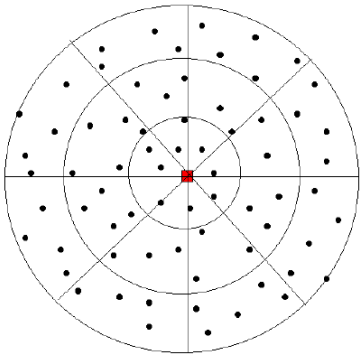


Figure 2.1: The first use of the corona model appears to be part of a clustering method which divides the network into coronas and wedges, with nodes being identified by their corona and wedge number [WOW+03].

identifying its corona and one identifying its wedge and the combination of these identities gives a nearly unique node ID if the number of coronas and wedges is large enough.

A variant of this method was used by later researchers as the basis for an examination of the energy hole problem [OS06], with the term *corona model* appearing to have first been used by Song *et al.* [SCL+08].

The energy hole problem is a special case of load imbalance in multi-hop wireless networks and forms the basis for this thesis. It is an imbalance inherent to the network, resulting from the design of the network and application. The energy hole problem was first formally analysed by Li and Mohapatra who used the corona model without naming it [LM05, LM07]. Their assumptions are the standard assumptions for the field and are quoted below with minor changes to notation and some explanatory notes in brackets:

1. In a clock-based many-to-one sensor network, each sensor node continuously generates constant bit rate (CBR) data (b bits/sec) and sends to a common sink through multihop shortest routes (either in terms of hops or physical distance).

2. Nodes are uniformly and randomly distributed, so that the node density, ρ is uniform throughout the entire network:

$$\rho = \frac{N}{A_{net}} \tag{2.1}$$

where N is the total number of sensor nodes and A_{net} is the coverage area of the sensor network.

3. All sensor nodes have the same, fixed transmission range of d meters.

4. Ideal MAC layer, i.e., transmission scheduling is perfect such that there are no collisions or retransmission.

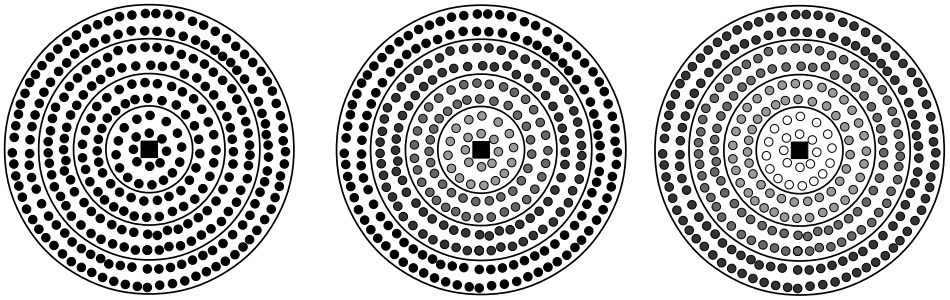
5. Sensor nodes use a location based greedy forwarding approach to transmit data packets to the sink. Quite a few such techniques have been proposed (for example, see [KK00]). In greedy forwarding, data packets are transmitted to a next-hop which is closest (physically) towards the destination.

6. Initially the network is well connected (meaning that every node has at least one path to the sink). The problem of what node density can ensure network connectivity is investigated by Bettstetter [Bet02].

In this thesis assumption 1 is made discrete such that nodes generate one data packet (of b bits) per round and rounds are long enough to ensure that all packets from all nodes are able to reach the sink before the next round starts.

The energy hole problem relates to the inherent bottleneck that is formed around the sink node because of multi-hop communication. The nodes that can communicate directly with the sink form the only link between the sink and the network and collectively forward all packets from the network. Their communication workload is much more than for other nodes and, assuming that all nodes starts with the same initial energy, they run out of energy first. When this happens no more packets can reach the sink and an energy hole is said to form. The formation of the energy hole is illustrated in Fig. 2.2.

The extent of the problem was shown analytically by Li and Mohapatra based on the assumptions. If there are k coronas each of width d, then the total network area, A_{net} is $\pi(dk)^2$. Assuming that there is no aggregation of packets, the number of packets that are forwarded collectively by the nodes in the inner-most corona is equal to the total number of bits generated by the network which is:



(a) (b) (c)

Figure 2.2: The energy hole forms over time from the imbalance in workload. Initially all nodes have the same energy reserves (a) but the nodes closer to the sink perform more work and deplete their batteries faster (b). Eventually, the nodes closest to the sink run out of energy and the sink is cut off from the network by the resulting energy hole (c).

$$A_{net}\rho b = \pi(dk)^2\rho b \tag{2.2}$$

If this work is evenly shared among the nodes in the inner-most corona, c_1, the workload of each node is:

$$\begin{aligned} L_1 &= \frac{\pi(dk)^2\rho b}{\pi d^2\rho} \\ &= k^2 b \end{aligned} \tag{2.3}$$

For all other coronas, the number of packets that need forwarding (including those generated by the nodes in the corona itself) is proportional to the network area outside the corona. The workload of each node in corona c_i:

$$\begin{aligned} L_i &= \frac{\pi\left((dk)^2 - (d(i-1))^2\right)\rho b}{\pi\left((di)^2 - (d(i-1))^2\right)\rho} \\ &= \frac{(k^2 - i^2 + 2i - 1)b}{2i-1} \end{aligned} \tag{2.4}$$

The ratio of the work performed by a node in the inner-most corona c_1 to those

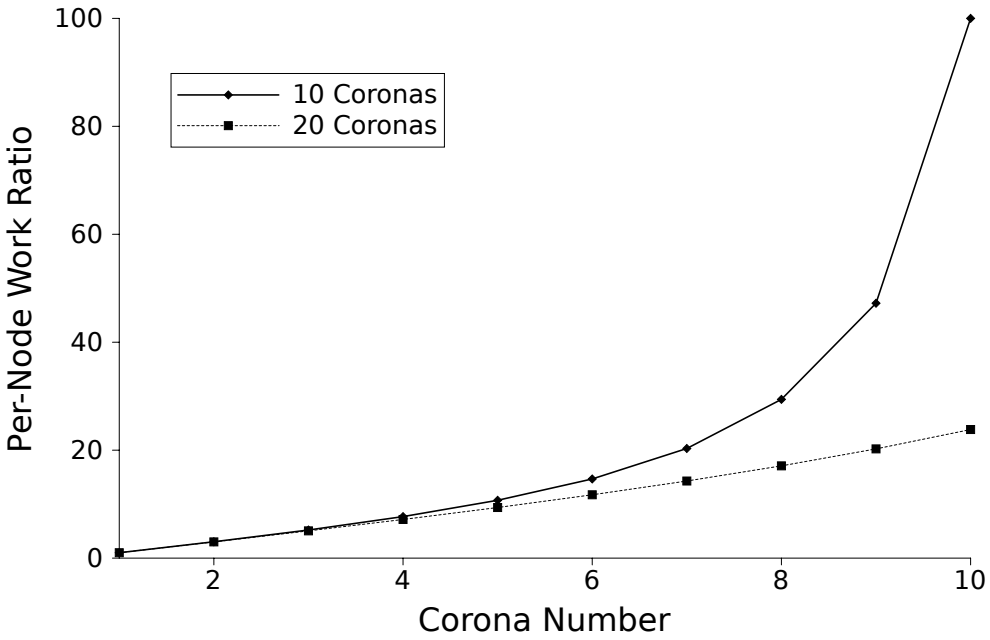


Figure 2.3: The work performed by each node in the inner-most corona is many times that of each node in coronas further out. The ratio grows polynomially but is significant even in the first few coronas.

in another corona c_i is:

$$\frac{L_1}{L_i} = \frac{(2i-1)k^2}{k^2 - i^2 + 2i - 1} \tag{2.5}$$

Equation (2.5) depends on both the number of coronas, k, and which corona number, i, is having its load compared to the inner-most corona but is independent of the node density and the data generation rate. It is therefore an inherent feature of the way the nodes are distributed in a network and gets worse as the network grows. Fig. 2.3 shows the ratio of work performed by each node in the inner-most corona compared to other coronas for a network with a total of ten and twenty coronas. The ratio grows polynomially with the corona number but is already significant at coronas close to the centre with each node in the inner-most corona performing more than three times the work of each node in the second corona in a ten corona network. In a larger network, for example one with twenty coronas, the proportional difference in work rates is less pronounced because the absolute total work is much greater. Nevertheless, it is clear from the results shown in Fig. 2.3 that the disparity in work rates is still significant.

The energy hole problem results in a very large wastage of energy. Lian *et al.* found that for networks with more than ten coronas as much as 90% of the initial energy of the network remains unused when the first nodes deplete their batteries [LNA05], although this will be even higher for very large networks. This observation has motivated extensive research into methods that can completely solve the energy hole problem and balance the energy consumption rates of all nodes, i.e. solving the inter-corona balance problem. However, in the next section I will show that all the potential solutions to this problem rely either on removing one of the initial assumptions or including another assumption that may not hold in all cases. It has already been shown that there are circumstances in which the energy hole problem cannot be solved [SNK05].

2.2 Solving the Energy Hole Problem

The purpose of this section is to highlight the conditions that must be met for inter-corona balance to be achievable and describe the reasons why these conditions may not be met by all networks. Inter-corona balance is complete load balancing in which all nodes in the network perform the same amount of work. Although it is phrased in terms of the corona model to contrast with other forms of load balancing it can be analysed and solved without reference to the corona model.

However, in order to solve the energy hole problem and produce inter-corona balance the network must have at least one of five constraints: perfect data aggregation, node mobility, transmission power control, clustering or non-uniform node distribution. Stojmenovic and Olariu have shown that in the absence of all these constraints it is impossible to solve the energy hole problem and achieve inter-corona balance [SO05].

2.2.1 Data Aggregation

Data aggregation is popular in sensor networks because it is a simple method for reducing energy consumption. The data generated by the sensors will often be highly correlated, therefore transmitting all the generated data would result in significant amounts of redundant or overlapping information. Data aggregation is designed to filter out some of this redundancy and reduce the total work that the network must perform which provides energy savings.

Krishnamachari *et al.* were among the first to analyse the potential energy savings from data aggregation by comparing the case of multi-hop communication with and without aggregation for a network in which only some of the nodes in the network generate data [KEW02]. The type of data aggregation they considered was such that relay nodes are able to combine multiple incoming packets into a single outgoing packet using functions such as `MAX`, `MIN` or `SUM`. This means that no matter how many incoming data packets a node receives, it only needs to transmit one towards the sink. Their results showed that using data aggregation could reduce energy consumption by between 50% and 80% for the scenarios they simulated.

Mhatre and Rosenberg generalised the notion of data aggregation by proposing a model for the relationship between the number of packets arriving at a relay node, x, and the number of packets it forwards, $\chi(x)$ [MR04a]:

$$\chi(x) = mx + c \tag{2.6}$$

They noted three general classes of aggregation. When $m = 0$ this corresponds to the type of aggregation assumed by Krishnamachari *et al.* in which there is only a single outgoing packet regardless of how many incoming packets there are. $m < 1$ indicates that there is some redundancy in the packets allowing for fewer outgoing packets than incoming ones but that the number of outgoing packets nevertheless increases with more incoming ones. Finally, $m = 1$ describes a network application which does not allow for any data aggregation. Buragohain *et al.* similarly divided aggregation into corresponding groups which they labelled *fully aggregated*, *partially aggregated* and *unaggregated*, though they did not propose a model for the amount of aggregation [BAS05].

Crucially for the purposes of inter-corona balance, Buragohain *et al.* showed that for fully aggregated networks any spanning tree provides inter-corona balance so long as the energy consumed when receiving a packet is zero (or negligible compared to the energy consumed transmitting a packet). However, they noted that the receive cost is usually not negligible and must be considered. In this case they proved that the optimal routing tree is a minimum degree spanning tree which is equivalent to minimising the number of children of each node in a static routing tree where every node has only one parent. Minimising the number of children also minimises the number of packets that a node receives, hence the energy consumption. However, generating a minimum degree spanning tree is an NP-Complete problem.

Although Buragohain *et al.* did not explicitly consider the energy hole problem, their results mean that data aggregation cannot usually be used to solve the problem because the number of children per parent has been shown by Macedo to fall according to the parent's corona number in networks with a uniform distribution of nodes [Mac09]. The average number of children per parent in corona c_i, C_i is given in equation (2.7) below. Since the number of children per parent cannot be made constant across the network, it is impossible to use data aggregation to

generate inter-corona balance unless the reception cost is so small that it can be ignored.

$$C_i = \frac{2i+1}{2i-1} \tag{2.7}$$

Furthermore, Mhatre and Rosenberg, argued that:

> "In most applications it may not be possible to fuse data from an arbitrary number of nodes into a single packet of fixed size. In general we expect the size of the aggregated data packet to increase with an increase in the number of input packets." [MR04a]

While perfect data aggregation can theoretically provide inter-corona balance in practice it cannot be relied upon. Not only is it unlikely to completely solve the energy hole problem because of reception costs but it also constrains the types of applications that the network can be used for.

2.2.2 Node Mobility

Mobility in wireless networks poses challenges because the links between nodes are continually changing. However, the changing of links can also be used to provide load balancing by rotating the set of nodes that form the gateway between the sink and the network. In theory, the sensor nodes could be moved around but since the nodes are resource constrained and the sink is not, it is usually the sink's movement that is assumed.

Wang *et al.* considered the question of sink mobility in the context of a square network with the nodes distributed in a grid and with a single sink that can move to share location with any of the sensor nodes [WBMP05]. The sink visits every point in the grid once for a varying period of time and they calculated how much time the sink should spend at each point, allowing for zero time to be spent at some positions.

To simplify the problem Wang *et al.* assumed that the sink can move from one position to another instantaneously and designed a linear program which takes as its inputs the power consumption rates for every node while the sink is at every potential position. The rates depend on the routing protocol that is used and

they considered a protocol in which the packet is routed along the perimeter of a rectangle connecting the source node and the sink as illustrated by Fig. 2.4. When the source node is not on the same row or column as the sink then two routes exist and packets are divided evenly between the two paths.

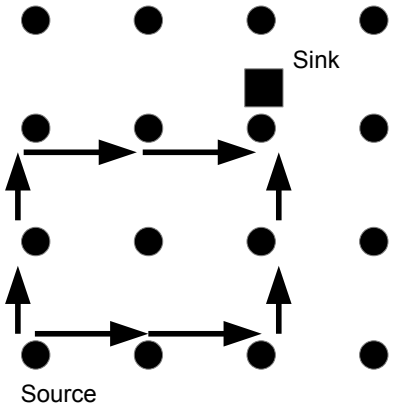


Figure 2.4: The routing protocol considered by Wang *et al.* finds the two paths that form a rectangle connecting the source node and the sink and divides the traffic flow evenly between them. Illustration adapted from [WBMP05].

Simulations results from Wang *et al.* found that the sink should spend the longest time in the corners of the network followed by some time in an inner square, as shown in Fig. 2.5. The pattern of the stops follows from the routing protocol and different routing choices would result in different stops.

Around the same time as Wang *et al.*, Luo and Hubaux were considering the same approach but for a circular network [LH05]. They first proved that, in terms of both latency and energy efficiency, the best single position for the sink is the centre of the network because as the sink moves away from the centre the maximum number of hops between a node and the sink increases. When the sink is at one edge of the network the worst case latency is double what it would be if the sink were at the centre. These extra hops not only increase latency but also result in more energy being consumed to forward packets to the sink.

However, Luo and Hubaux confirmed that the static sink causes imbalance in the work load and therefore that mobility can extend the lifetime of the network. They examined the question of what the optimum mobility strategy would be for a circular network and concluded that it was for the sink to move along the outer circumference of the network which, according to their simulation results, would

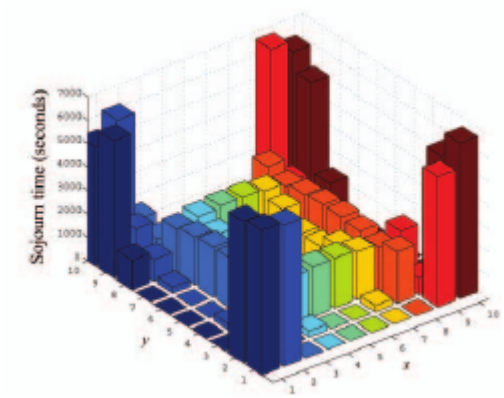


Figure 2.5: With the routing protocol used by Wang *et al.*, a mobile sink should spend the largest time in the corners and an inner square in order to maximise the lifetime of the network. Figure taken from [WBMP05].

reduce the workload of the heaviest loaded node by about 80% compared to a network with a static, central sink.

Basagni *et al.* argued that centralised approaches (such as the linear program of Wang *et al.*) are too costly both in terms of computation time and energy usage to be feasible in wireless sensor networks [BCM+08]. They therefore proposed a distributed method for controlling the mobility of the sink which they called Greedy Maximum Residual Energy (GMRE) in which the sink is effectively "drawn" to energy rich areas of the network. The network contains a number of sink sites arranged in a regular grid and the sink appoints nodes close to the sites to act as *sentinels* that monitor the available energy of the area and keep the sink informed. The sink can then make decisions about whether an alternative site has more energy available than its current one. Simulation results show that this technique can increase network lifetime by up to 350% compared to a static sink.

Yun and Xia [YX10] added another facet to the discussion by investigating a delay-tolerant network in which nodes are able to store a certain number of packets before forwarding them to the sink. This allows each node to wait until the sink is closer to it before sending its packets. They found that the network lifetime increased linearly with the number of sink locations and, in the best case, every node can delay its transmissions until the sink is in direct transmission range. In this case there is no need for routing or relaying and inter-corona balance is achieved as every node only transmits packets which are generated locally and at

the same rate for all nodes.

While sink mobility is a powerful method for maximising inter-corona balance there are potentially large costs associated with making the sink mobile. These energy costs have not been considered in the works cited in this section. Moreover, some terrains make a mobile sink (at least on the ground) extremely difficult. Finally, controlling the sink's mobility requires global knowledge of the network and significant overhead in terms of communication between nodes. For these reasons, sink mobility is not always a viable solution to the inter-corona balance problem.

2.2.3 Transmission Power Control

Transmission power control is the ability of nodes to fine tune the amount of power they put into their transmitted signals in order to control their range and the amount of power consumed during transmission. Inter-corona balance may be possible by tuning the transmission power of nodes according to their workload so that the heavier loaded nodes use a lower transmission power than the lighter loaded nodes.

Perillo *et al.* were among the first to investigate transmission power control as an approach to inter-corona balance [PCH04]. They made the simplifying assumption that every node could transmit directly to every other node and that the energy consumed in doing so was proportional to the square of the distance between the two nodes. A linear program was designed to calculate the traffic flows between nodes that maximised lifetime and simulation results showed that the approach could result in perfect inter-corona balance. They found that in order to balance the energy consumption rates across the network, the nodes furthest from the sink would occasionally transmit directly to it even though, as they stated, this involved "gross energy inefficiencies". Perillo *et al.* concluded that there was an:

> "inability to make good use of the energy of nodes furthest from the base station, even when utilizing the optimal distribution. Thus, even under the most ideal scenario (e.g., unlimited transmission ranges), varying the transmission power of individual nodes cannot alone solve the hot spot problem." [PCH04]

The approach of allowing nodes to switch between direct transmission to the sink and multi-hop communication was also considered by Guo *et al.* [GLW03], Liu *et al.* [LXG05] and Efthymiou *et al.* [ENR06]. All found similar results to Perillo *et al.* that balance was achieved when the nodes furthest from the sink occasionally transmitted directly to it even though this was very inefficient. The implication is that while this method can provide inter-corona balance the overall network lifetime is reduced (or at least not much extended) because of the inefficient use of the available energy.

Olariu and Stojmenovic investigated whether the energy hole problem could be solved through transmission power control even when most nodes were not able to transmit directly to the sink [OS06]. They assumed that every node had some maximum transmission range but that they could transmit any distance up to that maximum and consume less energy by transmitting shorter distances. They made use of the corona model and assumed that all nodes within the same corona transmit the same range which is enough to reach only the next corona. This reduces the question of inter-corona balance to tuning the width of each corona.

They proved that the most energy efficient form of multi-hop communication is for every hop to be the same length which explains why using transmission power control to balance energy consumption comes at the cost of efficiency. Nevertheless, Olariu and Stojmenovic developed an iterative algorithm for determining the transmission ranges (corona widths) that would balance the energy consumption. As with the previous attempts, they were able to balance the energy usage across the network but at the cost of energy efficiency.

One major problem with solutions relying on transmission power control is that they are unlikely to scale. Solutions similar to that proposed by Perillo *et al.* cannot scale because all nodes must be able to communicate directly with the sink. The solution of Olariu and Stojmenovic also has scalability issues because it will stop producing balance with a large number of coronas. This is because the outer coronas cannot be wider than the maximum transmission range and when they cannot be widened the balance breaks down.

Another problem is that these solutions assume that the radios onboard the sensor nodes can be finely tuned which is not normally the case. Most low power transceivers are able to select from a predetermined list of transmission power

settings only and often the list is short. For these reasons, transmission power control is not always a viable solution to the inter-corona balance problem.

2.2.4 Clustering

One method of achieving inter-corona balance is to divide the network into clusters so that nodes send their packets to their nearest cluster-head and the cluster-heads transmit packets to the sink. This creates a hierarchical network in which a node's workload depends on its role rather than its position. In effect multi-hop communication has been replaced with two stages, one stage from each node to cluster-head and a second from the cluster-head to the sink. Balance is achieved either by rotating the roles (in homogeneous networks) or by giving the cluster-heads more energy to begin with (in heterogeneous networks).

One of the earliest clustering solutions was proposed by Heinzelman *et al.* for homogeneous networks, called low-energy adaptive clustering hierarchy (LEACH) [HCB02]. In LEACH, the network lifetime is divided into rounds and in each round every node independently decides to become a cluster-head with a probability calculated locally based on the node's available energy. The cluster-heads advertise their status and for that round nodes that are not cluster-heads transmit to their nearest cluster-head which then transmits directly to the sink. By selecting the probabilities appropriately the nodes deplete their batteries at the same rate over time because they consume little energy transmitting the short distance to the cluster-head but occasionally consume more transmitting to the sink. A similar scheme was proposed by Lindsey and Raghavendra [LR02] in which a single cluster-head is chosen per round and nodes use multi-hop communication with perfect aggregation to reach it. The multi-hop communication within the cluster is designed to ensure that nodes further from the cluster-head do not consume more energy than other nodes inside the same cluster when the clusters are large and distance becomes an important factor.

Mhatre and Rosenberg considered the differences between the homogeneous and heterogeneous clustering approaches [MR04b]. They argued that in homogeneous networks, clustering can guarantee inter-corona balance (they did not use this expression) but that because all nodes must also act as cluster-heads, the hardware complexity and cost of the nodes increases. On the other hand, for heterogeneous

clustering, only the cluster-heads need to have complex hardware which allows the rest of the nodes to be simpler and cheaper. However, they pointed out that there will be energy imbalance inside each cluster because the nodes further away from the cluster-head need to use more energy to transmit to it than nodes closer. In effect clustering can result in the same energy hole problem but on a reduced scale.

For clustering to result in inter-corona balance in homogeneous networks the nodes must all be able to transmit directly either to the sink or to a cluster-head which limits the size of the network. If some nodes cannot do so then multi-hop communication is required again and the energy hole problem returns, albeit on a smaller scale. Even in heterogeneous networks the same issue applies because the cluster-heads themselves must all be able to directly transmit to the sink or else the energy hole problem exists among the cluster-heads. Furthermore, in heterogeneous networks some care must be taken to ensure that the relatively small number of cluster-heads end up in the right locations so that parts of the network are not disconnected because they have no cluster-head nearby.

2.2.5 Non-Uniform Node Distribution

A seemingly obvious solution to the build-up of work towards the centre of the network is to increase the resources in those areas in proportion to the work by deploying more nodes towards the centre. The first work in non-uniform distribution solutions is from Lian *et al.* who analysed a rectangular network with the sink placed in the middle of one edge [LNA05] (although all subsequent analysis has been based on the corona model). Fortunately, the approach of Lian *et al.* can be simply translated into the widely used corona model.

Lian *et al.* calculated how many nodes needed to be deployed in each part of the network by first dividing the network into sections and calculating the energy consumption rates for nodes in each section based on the number of packets flowing through that section. For a given desired lifetime it is simple to calculate how many nodes are required for each section given the section's energy consumption rate.

A similar analysis was carried out by Stojmenovic and Olariu using the corona model [SO05]. They showed that the required density of nodes in corona c_i, ρ_i,

relative to the density of the outer-most corona ρ_k, is given by equation (2.8) where k is the number of coronas. The same equation was derived independently by Liu *et al.* [LNN06].

$$\rho_i = \rho_k \frac{k^2 - (i-1)^2}{2i - 1} \tag{2.8}$$

These analyses all assumed that the energy consumed when receiving a packet was negligible and that data generation rates were unaffected by the addition of extra nodes so that if the number of nodes in a corona is doubled compared to the uniform distribution then each node generates packets at half the rate. Wu *et al.* analysed the problem without these assumptions and found that, because the nodes in the outer-most corona do not have to forward any packets, it is impossible to achieve perfect inter-corona balance [WCD08]. However, inter-corona balance can be achieved among all coronas except the outer-most by deploying nodes according to a geometric progression with common ratio $q > 1$. He and Xu maintained the assumption of constant data-generation rates but included the energy consumption from reception in their analysis [HX10]. They concluded that perfect inter-corona balance was possible if the number of nodes per corona (including the outer-most corona), N_i, was as specified in equation (2.9) where e_{tx} and e_{rx} are the energy consumed during transmission and reception of a packet respectively.

$$N_i = \pi\rho_k \left(k^2 - (i-1)^2\right) + \pi\rho_k \frac{e_{rx}}{e_{tx}} \left(k^2 - i^2\right) \tag{2.9}$$

Table 2.1 shows the number of nodes that would be deployed according to the different solutions mentioned in this section compared against a simple uniform distribution of nodes. It is evident that the cost of achieving high inter-corona balance using this method is very high. The solution with the fewest required nodes (Stojmenovic and Olariu) still requires ten times as many nodes as the uniform distribution which makes these solutions very expensive. What is more, the inner-most corona which has the smallest area requires the largest number of nodes and it would be difficult to accommodate such a large density. A final problem is that the nodes would have to be deployed with considerable control over their final positions in the network and this may not be feasible in harsh terrains or with very large networks.

Table 2.1: The non-uniform distribution solution requires a large number of extra nodes in order to balance the energy usage.

Corona number	Uniform (ρ_k=1)	Stojmenovic and Olariu [SO05]	Wu *et al.* (q=2) [WCD08]	He and Xu [HX10]
1	3	707	745,472	1,467
2	9	704	372,736	1,454
3	16	694	186,368	1,427
4	22	679	93,184	1,388
5	28	657	46,592	1,335
6	35	628	23,296	1,270
7	41	594	11,648	1,191
8	47	553	5,824	1,099
9	53	506	2,912	994
10	60	452	1,456	877
11	66	393	728	746
12	72	327	364	602
13	79	254	182	444
14	85	176	91	274
15	91	91	91	91
Total	**707**	**7,415**	**1,490,944**	**14,659**

2.2.6 Summary

The previous section has shown that solving the energy hole problem and achieving inter-corona balance requires imposing constraints on the network and making assumptions that sometimes do not hold. Table **??** summarises the solutions discussed and highlights two of the constraints each solution imposes. It is clear that for a very large group of sensor networks, there is no solution to the energy hole problem and inter-corona imbalance is unavoidable. For these networks one of the other form of load balancing should be maximised and it has already been proved that in terms of network lifetime, maximising inner-corona balance is as optimal as both of the other two types of balancing.

2.3 Dynamic Routing

In this thesis the focus is on constructing static routing trees in which nodes determine their parent once for the entire lifetime of the network or at least for a significant period of time. An alternative, however, is dynamic routing in which nodes maintain a list of potential parents and select between them on-the-fly based on some cost function. This method is often required for networks with fast changing properties, for example mobile nodes, but comes at the cost of additional overhead as will be discussed at the end of this section.

Shah and Rabaey were one of the first to propose a dynamic routing scheme, called Energy Aware Routing (EAR) [SR02]. The underlying principle behind EAR is that each node constructs and maintains a routing table with a list of their potential parents, a path cost associated with each one and the probability of forwarding to that parent based on the path cost. The path cost is calculated as the potential parent's path cost plus an energy metric which is the weighted product of the residual energy of the potential parent and the energy consumed when transmitting to it. For each packet that needs forwarding a potential parent is selected at random according to the probabilities in the routing table so that nodes which are more heavily burdened and therefore have lower residual energy are chosen as parents less often than nodes with more residual energy. Periodically, nodes broadcast beacons containing estimates of their residual energy so that costs and probabilities can be kept fresh.

Many other dynamic schemes have been proposed that all work along similar lines to EAR. Puccinelli and Haenggi proposed a routing protocol, Arbutus, which calculates a parent's path cost based on the quality of the link to the parent, the minimum quality of the parent's path and a measure of the path's workload [PH08, PH09]. Periodic beaconing is required with Arbutus (as with every dynamic routing scheme) but no routing table is maintained. Instead, the beacons are used to select the parent with the lowest path cost and that parent is used exclusively until a beacon is received indicating that a different parent now has a lower path cost. A similar scheme was proposed independently by Daabaj *et al.* [DDK09a, DDK09b].

Tellioglu and Mantar argued that dynamic routing schemes should not select different parents with a given probability; rather they should divide the traffic

flow between all potential parents in different proportions so that parents with lower costs are sent a larger proportion of the total traffic flow [TM09]. They proposed a Proportional Load Balancing (PLB) scheme in which the path costs are used to define the proportion of traffic that should be forwarded to each parent. PLB has the advantage of reducing the amount of change from round to round under dynamic routing.

As mentioned, the drawback with all dynamic schemes is the need for every node in the network to periodically broadcast a control packet in order for nodes to keep their neighbourhood information up to date. These packets must be sent throughout the entire lifetime of the network and represent a significant overhead especially in sensor networks where control packets are likely to be about the same size as data packets [BBB09]. In the types of networks considered in this thesis, the network conditions remain stable for extended periods of time and therefore a static routing tree is viable. For networks where static trees are viable it is sensible to avoid dynamic routing schemes in order to avoid the significant overhead that comes with them.

2.4 Degree Balancing

Degree balance is a form of intra-corona balance concerned with minimising the variation in the number of children that nodes within the same corona adopt. If the variation can be reduced to zero in all coronas then intra-corona and inner-corona balance are also achieved because if all nodes have the same number of children and all those children have the same number of children etc. then all nodes have the same number of descendants and therefore load.

Andreou *et al.* proposed an algorithm that could take an arbitrary routing tree and convert it into a degree balanced tree. They noted that if all nodes are within communication range of each other then balancing algorithms such as AVL Trees or B-Trees could be used to generate a fully balanced tree. Moreover, if the depth of the final routing tree were Δ then every node would have approximately the same number of children, defined in equation (2.10) as the branching factor, Φ where N is the number of nodes in the network. Their starting point was that if every node in a balanced tree has Φ children then the tree would be of depth $\log_\Phi N$.

$$\Phi = \sqrt[\Delta]{N} \tag{2.10}$$

Andreou *et al.* suggested that in a realistic network, where nodes cannot communicate directly with all other nodes, a near-balanced tree could be constructed in which the number of children adopted by each node was at most equal to the branching factor Φ. They called their proposed algorithm Energy-Driven Tree Construction (ETC) and it is a degree balancing algorithm (even though they do not refer to it as such) because it aims to minimise the variation in the number of children adopted by nodes in the same level of the routing tree without concern for overall loads.

The ETC algorithm is supervised by the sink and initially constructs a shortest-path routing tree. The sink queries this tree to discover the number of nodes and the depth of the tree in order to calculate the branching factor using equation (2.10). This is flooded through the network and nodes use it to rebalance their subtrees by instructing some of their children to switch to new parents. Although Andreou *et al.* describe their algorithm as distributed, a distinction can be drawn between a routing protocol in which nodes decide for themselves who to forward packets to using only information from a one-hop neighbourhood and protocols in which this decision is made by other nodes using information gathered from a wider area. In this thesis the first type of protocol is referred to as fully distributed and the second as *supervised*. With this distinction, ETC would be considered a supervised routing protocol.

The main functionality of ETC is in response to a node receiving the branching factor through a **newParent** packet and is specified in algorithm 2.1. For ETC to work every node must gather a list of alternative parents (APL) from each of its children so that it can determine which of its children can switch parents and to which other node.

When a **newParent** packet is received the node examines whether it has been assigned a new parent or not and switches if it has. It then moves on to balancing its own children by comparing the number of children it currently has to the branching factor. If it has too many children it iterates through them looking for a child which has alternative parents available and instructs that child to switch parent. It continues instructing children to switch until it has Φ children

Algorithm 2.1 ETC Balancing

```
1: function newParent(Φ,newParentID)
2:     if newParentID != NULL then          ▷ Switch to newParent if specified
3:         parent = newParentID
4:     end if
5:     while children.size() > Φ do
6:         for all child ∈ children do
7:             if APL(child).size() > 0 then
8:                 altParent = APL(child).get(random())
9:                 send newParent(Φ,altParent) → child
10:                children.remove(child)
11:            else
12:                send newParent(Φ,NULL) → child
13:            end if
14:        end for
15:    end while
16: end function
```

or fewer. The authors suggest that if a child attempts to switch to a parent that cannot adopt it without having more than Φ children, then the child abandons the switch and informs its current parent of the failure. Another alternative parent can be chosen for the child if one is available and if none is, the child cannot be switched.

A fully distributed algorithm was proposed by Huang *et al.* who took a node's degree as a proxy for its energy consumption rate [HCWC09]. Their algorithm, MBT (for Minimum Balanced Tree), starts from scratch to create a balanced tree rather than attempting to rebalance an existing tree like ETC does. Initially, all nodes set their *height* in the tree to ∞ except the sink which sets its *height* to zero. During a construction phase each node will periodically "explore" its neighbourhood by querying its neighbours to gather information about their *height* and *degree*. As specified in algorithm 2.2, each node primarily selects the neighbour with minimum height as its parent but where more than one neighbour has the same height then the one with minimum degree is chosen. If a new parent is chosen with a different height then the node floods its descendants informing them of the height change.

In the MBT algorithm, if a node finds an alternative parent with fewer children

Algorithm 2.2 MBT Explore

```
1: function EXPLORE(neighbours)
2:     for all neighbour ∈ neighbours do
3:         if my.height() > neighbour.height()+1 then
4:             parent = neighbour
5:             my.height = neighbour.height()+1
6:             floodSubtree(my.height())
7:         else if my.height() == neighbour.height()+1 then
8:             if parent.degree()-1 > neighbour.degree() then
9:                 parent = neighbour
10:            end if
11:        end if
12:    end for
13: end function
```

then it will switch to become that node's child. The result is that, in the final routing tree, the difference between the maximum and minimum number of children of nodes in each level should be at most one. However, MBT requires significant exchange of control packets as nodes explore their neighbourhood numerous times. Furthermore, the algorithm will take some time to converge to a stable routing tree although Huang *et al.* did not analyse the conditions and number of explorations required for convergence.

Chatzimilioudis *et al.* proposed a degree balancing algorithm with lower overhead and definite convergence but with potentially poorer performance [CZYG10]. Their protocol, MHS (Minimum HotSpot Query Routing Tree), works by having nodes make their parent selection sequentially so that the nodes can become aware of new adoptions and have a more up-to-date picture of the degree of their potential parents when they make their selection.

The algorithm works in rounds and in each round the nodes that joined the tree in the previous round are the parents and the nodes in direct communication with the parents that are not part of the tree are the children. In the first round the sink is the only parent and all nodes in direct communication with the sink are the children. At the beginning of each round, the parent nodes broadcast beacons which are collected by the child nodes who construct potential parent lists. Initially, none of the parents has any children so their degrees are all the same (one). The child nodes wait for a period of time after receiving their first beacon during which they listen out for more beacons from other potential

parents. The time they wait must be long enough to ensure that they hear from all their potential parents.

By the end of this period, every child node has a list of all its neighbours that could act as its parent. It must then decide which neighbour to choose. In order to maximise degree balance, the nodes should choose the neighbour with the fewest children but at first none of the neighbours has any children. It is only as the parents adopt children that disparity grows and therefore nodes should ideally wait until all other children have been adopted before making their choice. Since this would clearly prevent any nodes joining the tree a compromise is found whereby nodes wait for different periods before making their choice.

Chatzimilioudis *et al.* noted that the order of selection has a significant impact on the performance and suggested that the best ordering is for the nodes with the fewest potential parents to select first. A node with only one parent choice will not gain from knowing the number of children adopted by other parents. However, nodes with many options can improve the balance significantly by making good choices based on more information. It therefore makes sense that the nodes with the fewest options should choose first because they cannot derive as much benefit from the extra information as the nodes with the most options. When two or more nodes have the same number of options then it does not matter which goes first and so a random back-off is used to avoid collisions. The time, t_{choose}, that a node should wait before selecting its parent is given in equation (2.11), where $|P|$ is the number of potential parents.

$$t_{choose} = timeslotsize * (|P| + timeslotsize * random(0, 1)) \tag{2.11}$$

When the time t_{choose} has elapsed for a given node it must choose a parent and it transmits an adoption request packet to its chosen parent which then broadcasts a confirmation. The confirmation is received by all nodes who have that parent in their potential parents list and they update its degree in preparation for selecting their own parent. After a certain amount of time long enough to ensure that all child nodes have made their selections and been adopted, the next round starts; the child nodes become the parent nodes and broadcast their own beacons starting the process of building up the next level of the routing tree.

In this way MHS is fully distributed and has significantly less overhead than

MBT because for each node's adoption only two packets are needed (and only one further packet is transmitted as a beacon). However, once a node has joined the tree it cannot switch parents and improve balance which it can do in MBT. These two distributed algorithms will be compared in Chapter 5 and the best performing will serve as a benchmark for the novel distributed protocols that I will propose.

2.5 Inner-Corona Balance

So far this review has shown that there are numerous proposed methods for inter-corona balance but that each group of solutions places constraints on the network such as requiring finely tuneable radios or a mobile sink. Without those constraints, distributed protocols have been described that are dynamic and therefore require significant overhead to maintain up-to-date routing tables or else aim only to maximise degree balance. In this section algorithms aiming to maximise inner-corona balance will be reviewed; all, bar one, are centralised algorithms that require accurate global knowledge and therefore do not scale well. The remaining algorithm is not fully distributed either and is of limited effectiveness. To the best of my knowledge, the two new protocols proposed in this thesis are the first fully distributed algorithms that aim to maximise inner-corona balance in a static routing tree.

Hsiao *et al.* were one of the first to propose an algorithm for load balancing in many-to-one networks in the context of wireless access networks [HHKV01]. Their algorithm takes an arbitrary routing tree and incrementally improves its balance. Central to the process is the balance index which is Jain's fairness index as defined in equation (2.12) where w_i is the load on subtree i and n is the number of subtrees [JCH84]. The index has the useful properties of being bounded between zero and one and being monotonic which allows it to be used in a greedy algorithm.

$$\theta = \frac{\left(\sum_{i=1}^{n} w_i\right)^2}{n \sum_{i=1}^{n} w_i^2} \tag{2.12}$$

The basic mechanism proposed by Hsiao *et al.* is to examine the neighbours of

each node and evaluate whether changing a node's parents would reduce the difference in load between its old and new subtrees. Such a reduction would increase the balance of the network. The algorithm is iterative with one node switching subtrees per iteration and they considered three heuristics for selecting the node to switch: best-first, random and weighted. The best-first heuristic examines all nodes and selects the switch which will have the biggest effect on balance. Random, as the name suggests, selects one of the possible switches randomly with equal probability while the weighted heuristic is also random but assigns a higher probability to switches that make bigger improvements to balance. The advantage of the random and weighted heuristics is that they use only local information, albeit in a supervised fashion, whereas best-first must have global knowledge. As a final point the authors noted that the greedy nature of the algorithm means that it could get stuck at a local maximum and therefore proposed to use simulated annealing as well. Algorithm 2.3 gives the pseudocode for one iteration of the best-first heuristic.

Hsiao *et al.* also proposed a distributed implementation of their algorithm but suggested that in order to avoid nodes using stale information and switches clashing, the process should be supervised by the sink node. In each iteration the network is queried to gather global information about the loads of all nodes and subtrees and the sink selects one subtree to consider for switching. Every node in the selected subtree considers, independently, whether and to what extent balance would be improved if it switched. The nodes that can switch inform the sink of how much balance could be improved by a switch and then the sink selects the best switch and instructs one node to switch. Although this is technically distributed there is nevertheless global gathering of information,extremely high overheads and the final routing decisions are not made locally so that this algorithm could be described as supervised according to the definition in Section 2.4.

Dai and Han extended the work of Hsiao *et al.* by proposing an algorithm for the construction of an initial, roughly balanced tree [DH03]. Their node-centric algorithm is centralised and iteratively builds a tree by attaching the heaviest loaded, unconnected node to the subtree with the lightest current load. The authors introduced the notion of *growth space* to act as a tie-breaker when multiple choices exist which is always the case when all nodes have the same weight. The

Algorithm 2.3 Hsiao *et al.*

```
1: function generateTree(nodes)
2:     currentBalance = balance(nodes)
3:     bestNode = NULL
4:     newParent = NULL
5:     bestBalance = currentBalance
6:     for all node ∈ nodes do
7:         nodeWeight = node.weight
8:         for all parent ∈ node.potentialParents() do
9:             currentWeight = node.subtree.weight()
10:            parentWeight = parent.subtree.weight()
11:            if parentWeight < currentWeight then
12:                nodesCopy = nodes
13:                altNode = nodesCopy.get(node.index)
14:                altNode.parent = parent
15:                altBalance = balance(nodesCopy)
16:                if altBalance > bestBalance then
17:                    bestNode = node
18:                    newParent = parent
19:                    bestBalance = altBalance
20:                end if
21:            end if
22:        end for
23:    end for
24:    if bestNode != NULL then
25:        bestNode.parent = newParent
26:    end if
27: end function
```

growth space of a node is defined as the sum of the number of unconnected (unmarked) neighbours of a node's unconnected neighbours, excluding common links as illustrated in Fig. 2.6. The concept behind the growth space is that subtrees should grow into areas where more nodes are unattached because this allows more choice in the future about which node to attach to which subtree.

The node centric algorithm produces a roughly balanced tree but not a fully balanced one. Dai and Han therefore proposed adopting the best-first rebalancing approach of Hsiao *et al.* for the final balancing element. In each iteration the heaviest loaded subtree is determined and the deviation between its current load and the optimal load is calculated. The subtree is then searched for the

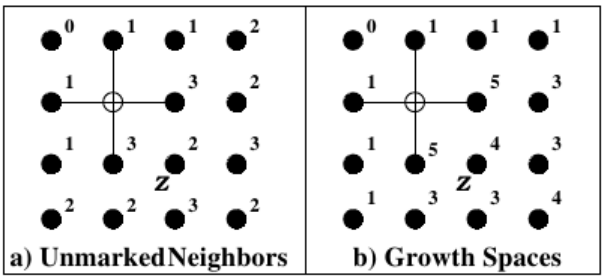


Figure 2.6: The number of unmarked neighbours (a) measures the number of neighbours that are unattached to the tree and is used to calculate the growth space (b) of a node which is the sum of the unmarked neighbours of a node's unmarked neighbours (excluding common links). Diagram taken from [DH03].

node whose load is most similar to the deviation that can be switched to a different subtree. This process continues until some stopping condition, typically a maximum number of iterations. Algorithm 2.4 gives the pseudocode for the construction of the initial tree starting after the inner-most nodes have already been adopted by the sink and the subtrees have been created. Algorithm 2.5 gives the pseudocode for the tree rebalancing part of their proposed solution. A nearly identical rebalancing algorithm was proposed by Chu *et al.* [CTL+09].

Buragohain *et al.* noted that finding the optimal routing tree for networks without aggregation is NP-Complete [BAS05]. This is because, if all nodes have the same initial energy and data generation rates, the problem of inner-corona balance is equivalent to the problem of constructing a capacitated spanning tree. A capacitated spanning tree is a spanning tree in graph theory in which the capacity of every subtree rooted at a level one node is less than or equal to a defined maximum. The capacity of a subtree is the total number of nodes in the subtree. Chandy and Lo studied the problem of capacitated spanning trees [CL73] and it was included in a list of NP-Complete problems by Garey and Johnson [GJ79].

The translation from capacitated spanning tree to inner-corona balance is straightforward. The nodes in level one of a tree are the nodes in the inner-most corona and the capacity of their subtrees is the number of descendants they have. If the maximum capacity is set correctly, then constructing a capacitated spanning tree is equivalent to creating a routing tree with inner-corona balance because all subtrees are the same size, ie all of the inner-most corona nodes have the same

Algorithm 2.4 Dain and Han, Generate Tree

```
1: function generateTree(nodes,subtrees)
2:     numberAssigned = subtrees.size()
3:     while numberAssigned < nodes.size() do
4:         lightestTree = NULL
5:         lightestLoad = nodes.size()
6:         for all subtree ∈ subtrees do
7:             if subtree.load < lightestLoad then
8:                 lightestLoad = subtree.load
9:                 lightestTree = subtree
10:            end if
11:        end for
12:        borderNodes = inRange(nodes,lightestTree)
13:        heaviestNode = NULL
14:        heaviestLoad = 0
15:        largestGrowthSpace = 0
16:        for all node ∈ borderNodes do
17:            if node.load > heaviestLoad then
18:                heaviestLoad = node.load
19:                heaviestNode = node
20:                largerstGrowthSpace = growthSpace(node,nodes)
21:            else if node.load == heaviestLoad then
22:                if growthSpace(node,nodes) > largestGrowthSpace then
23:                    heaviestNode = node
24:                    largestGrowthSpace = growthSpace(node,nodes)
25:                end if
26:            end if
27:        end for
28:        heaviestNode.subtree = lightestTree
29:        numberAssigned++
30:    end while
31: end function
```

Algorithm 2.5 Dain and Han, Rebalance Tree

```
1: function RebalanceTree(nodes,subtrees,maxIterations)
2:     averageSize = nodes.size()/subtrees.size()
3:     numberIterations = 0
4:     while numberIterations < maxIterations do
5:         heaviestTree = NULL
6:         heaviestLoad = 0
7:         for all tree ∈ subtrees do
8:             if tree.load > heaviestLoad then
9:                 heaviestTree = tree
10:                heaviestLoad = tree.load
11:            end if
12:        end for
13:        deviation = heaviestLoad - averageSize
14:        bestNode = NULL
15:        nodeDeviation = deviation
16:        for all node ∈ heaviestTree do
17:            if —node.load - deviation— < nodeDeviation then
18:                bestNode = node
19:                nodeDeviation = —node.load - deviation—
20:            end if
21:        end for
22:        bestNode.switch()
23:    end while
24: end function
```

number of descendants.

Buragohain *et al.* went on to propose a centralised algorithm, Energy Conserving Routing Tree (ECRT), for the construction of an approximately-balanced tree by considering the lifetime of the tree. The algorithm iteratively grows the routing tree by adding the node that minimises the reduction in network lifetime, as described in algorithm 2.6. They noted, though, that the tree constructed was not optimal and could be improved. They therefore proposed a tree rebalancing algorithm that could be used independently of ECRT or after it which would attempt to maximise network lifetime. They defined a locally optimal tree as one in which no nodes could be switched to a different parent to improve network lifetime. With that definition they proposed an algorithm, LOCAL-OPT, that would sequentially examine each node to determine whether switching its parent would increase lifetime and make the switch if it would. The algorithm, specified in algorithm 2.7, continues until no more switches can be made.

To the best of my knowledge the only existing distributed routing algorithm that aims to maximise inner-corona balance was proposed by Chen *et al.*, called adjustable converge-cast tree (ACT) [CTC10]. The aim of the algorithm is to rebalance a shortest path tree starting at the leaf nodes and working towards the sink. The rebalancing is done by nodes (known as grandparents) instructing their grandchildren to switch from being children of one of the grandparent's children (known as the parents) to a different child of the grandparent. The relationship between grandparents, parents and children is illustrated in Fig. 2.7.

The rebalancing is initiated by the reception of one *tree reply* packet, **TREP** from every one of the parents. When a node fails to adopt any children during the tree construction phase it immediately transmits a **TREP** packet to its parent. Other nodes transmit **TREP** packets after they have completed their own rebalancing. When a grandparent has received one **TREP** packet from every one of its children (the parents) it starts the rebalancing by utilising the information it has gained from the packets to calculate is load-balancing factor (LBF). The LBF is the min/max ratio between the smallest subtree rooted at one of the parents and the largest such subtree and is a measure of the balance of the subtree rooted at the grandparent itself.

If a grandparent's LBF is equal to one then its subtree is perfectly balanced and the node transmits its own **TREP** packet to its parent (what we might call the

Algorithm 2.6 ECRT

```
1: function ECRT(nodes,sink)
2:     numAttached = 0
3:     while numAttached < nodes.size() do
4:         borderNodes = inRange(nodes)
5:         bestLifetime = 0
6:         bestNode = NULL
7:         bestParent = NULL
8:         for all node ∈ borderNodes do
9:             bestNodeLifetime = 0
10:            bestNodeParent = NULL
11:            for all parent ∈ node.potentialParents do
12:                if parent.attached() then
13:                    altNodes = nodes.copy()
14:                    altNode = altTree.get(node.name)
15:                    altNode.parent = parent
16:                    altLifetime = lifetime(altNodes)
17:                    if altLifetime > bestNodeLifetime then
18:                        bestNodeParent = parent
19:                        bestNodeLifetime = altLifetime
20:                    end if
21:                end if
22:            end for
23:            if bestNodeLifetime > bestLifetime then
24:                bestLifetime = bestNodeLifetime
25:                bestNode = node
26:                bestParent = bestNodeParent
27:            end if
28:        end for
29:        bestNode.parent = bestParent
30:        numAttached++
31:    end while
32: end function
```

Algorithm 2.7 LOCAL-OPT

```
1: function LOCAL-OPT(nodes)
2:     done = false
3:     while ! done do
4:         done = true
5:         currentLifetime = lifetime(nodes)
6:         for all node ∈ nodes do
7:             for all parent ∈ node.potentialParents do
8:                 altNodes = nodes.copy
9:                 altNode = altNodes.get(node.name)
10:                altNode.parent = parent
11:                altLifetime = lifetime(altNodes)
12:                if altLifetime > currentLifetime then
13:                    node.parent = parent
14:                    done = false
15:                end if
16:            end for
17:        end for
18:    end while
19: end function
```

great-grandparent). However, if the LBF is greater than one then the grandparent tries to rebalance the load among the parents. A special case exists where the difference between the largest and smallest subtrees is exactly one in which case it will be impossible to balance the load because moving one grandchild from the largest to the smallest subtrees will simply make the two subtrees switch places without affecting the LBF.

In order to balance the load, the node examines those of its grandchildren that are children of the largest subtree to determine whether they have an alternative parent in the smallest subtree. If they do then the change is recorded and the LBF is updated. The node continues examining grandchildren until no more improvements can be made. At this point it broadcasts a *tree adjustment*, **TADJ**, packet that contains the list of switches. The children that receive the packet examine it and if one of their children is listed as requiring a switch they forward the packet on to the relevant nodes. The complete algorithm is specified in algorithm 2.8.

The ACT algorithm as proposed by Chen *et al.* only allows nodes to move from the largest to smallest subtrees although the min/max ratio may be improved

Algorithm 2.8 ACT

```
1: function ACT(children,grandchildren)
2:     done = false
3:     while !done do
4:         done = true
                                                    ▷ Find Current LBF
5:         largestSubtreeSize = 0
6:         largestSubtree = NULL
7:         smallestSubtreeSize = ∞
8:         smallestSubtree = NULL
9:         for all child ∈ children do
10:            if child.load > largestSubtreeSize then
11:                largestSubtree = child
12:                largestSubtreeSize = child.load
13:            end if
14:            if child.load < smallestSubtreeSize then
15:                smallestSubtree = child
16:                smallestSubtreeSize = child.load
17:            end if
18:        end for
19:        LBF = smallestSubtreeSize / largestSubtreeSize
                                          ▷ Find Switches That Improve LBF
20:        if LBF < 1 && largestSubtreeSize - smallestSubtreeSize > 1 then
21:            switchedGranchildren = {}
22:            for all grandchild ∈ largestSubtree do
23:                for all parent ∈ grandchild.potentialParents do
24:                    tmpMin = smallestSubtreeSize+grandchild.load
25:                    tmpMax = largestSubtreeSize-grandchild.load
26:                    tmpLBF = tmpMin/tmpMax
27:                    if tmpLBF > LBF then
28:                        LBF = tmpLBF
29:                        switchedGranchildren.add(grandchild)
30:                        done = false
31:                    end if
32:                end for
33:            end for
34:            send TADJ(switchedGrandchildren)
35:        end if
36:    end while
37: end function
```

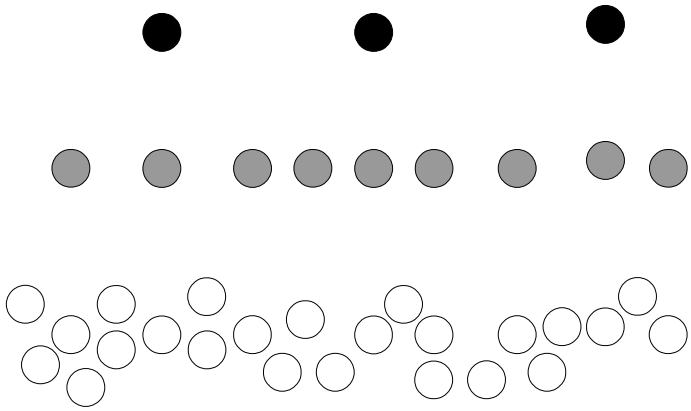


Figure 2.7: The ACT algorithm involves three levels of the routing tree working together. The grandparents (black nodes) instruct the grandchildren (white nodes) to switch from one parent (grey nodes) to another in order to maximise balance.

by moving nodes away from the largest subtree even if they do not move to the smallest or by moving nodes from some other subtree to the smallest one. This modification is trivially done. However, a more significant problem is that ACT only allows a node to move its grandchildren from one child's subtree to another which means that each node's number of descendants remains unchanged by the load balancing process. Only the root node's implementation of algorithm 2.8 will affect the inner-corona balance by moving nodes from one subtree to another but the root can only move level two nodes together with all their descendants which means that small changes in balance are hard to make. Moreover, the overhead of ACT is very high because large amounts of information need to be collected and passed through two hops for the load balancing and then switch instructions must be passed two hops back down again.

2.6 Summary and Conclusions

The purpose of this review has been to show the need for new protocols that are fully distributed, create a static routing tree and aim to maximise inner-corona balance. Although inner-corona balance is the least extensive of the four types of balance defined in Chapter 1, it has important advantages that were described in Section 1.1. The first part of this review proved the earlier claim

that inter-corona balance was only achievable by imposing various constraints on the network. The proposed methods for achieving inter-corona balance and some of the corresponding constraints were summarised in Table **??**.

This review also considered dynamic routing protocols and described how they result in increased communication overhead which is why they are best avoided when not needed. Since the networks considered in this thesis are stable for long periods of time, dynamic routing is not needed and therefore it is best to create a static routing tree and avoid the overhead.

By describing the proposed protocols that focus on degree balancing it was shown that this type of load balancing is achievable with distributed algorithms. For this reason, the degree balancing protocols can serve as a benchmark for the novel distributed algorithms that will be proposed later in this thesis.

The final part of this review demonstrated that the problem of maximising inner-corona balance has been studied before. Centralised algorithms have been proposed to solve the problem but such algorithms require gathering global knowledge which is expensive in terms of energy usage and restricts scalability. One attempt has been made to solve the problem in a distributed fashion but the proposed solution has significant drawbacks that prevent it from maximising inner-corona balance.

It is clear that there is a need for new protocols because the existing literature does not contain distributed algorithms that maximise inner-corona balance with a static routing tree. The primary aim of this thesis is the proposal of new protocols that do so which will be done in Chapters 6 and 7. However, before that the assumptions and metrics used in the rest of the thesis are detailed in the next chapter.

Chapter 3

Assumptions and Metrics

The purpose of this chapter is to provide a complete list of the assumptions that underpin this thesis and the metrics that will be used to measure protocol performance. The assumptions are all commonly used in the field but nevertheless a brief description and justification is provided for each one. As part of the justification for using the unit disk graph model, energy-efficient position-based routing is revisited and it is shown that absolute reception-based blacklisting is the optimal strategy. The metrics used to evaluate and compare the performance of different routing algorithms is discussed at the end of this chapter along with a short description of the major statistical measures used throughout the thesis.

3.1 Assumptions

For the most part, this thesis follows the assumptions made by Li and Mohapatra in their analysis of the energy hole problem [LM05, LM07]. The full list of assumptions in this thesis is as follows:

1. There is a circular network of radius R.
2. There is a single, resource-unconstrained, central sink.
3. All nodes are static.
4. Nodes are uniformly and randomly distributed such that the network density, ρ, is constant across the network area.

5. The network is homogeneous and all sensor nodes have the same initial energy capacity.
6. The network is dense enough to ensure connectivity.
7. The network lifetime is divided into fixed length rounds long enough to ensure that all packets generated during that round can reach the sink.
8. Every node generates one fixed size data packet per round.
9. Routing is through multi-hop communication.
10. All links have enough capacity to transfer the required data.
11. There is no data aggregation.
12. The MAC algorithm is ideal and minimises collisions and retransmissions.
13. Transmission range follows the unit disk graph model.

The mathematical model resulting from these assumptions certainly does not fit well for all types of sensor networks. However, for the kind considered in this thesis it is a good fit. The typical application considered in this thesis is some form of environmental monitoring over a large and relatively regularly shaped area. Therefore, almost all assumptions made are entirely appropriate. Clearly, the MAC algorithm in use will not be ideal but whatever errors it contains will likely affect all routing layer protocols equally or in proportion to the number of packets they require which means that it can be isolated from the routing layer without affecting the correctness of any comparisons. Similarly, in the real world the unit disk graph model does not hold but its use in simulations comparing the performance of protocols is strongly justified in this chapter.

3.1.1 Circular Network

A circular network is part of the widely used corona model that was introduced in the previous chapter. One reason for assuming a circular network is that it facilitates mathematical analysis; however, it also flows from the use of the unit disk graph (UDG) model. Under the UDG model every node, including the sink, has the same fixed transmission range surrounding it creating a circular *reachable area*. Therefore, the children of the sink form a circle around it. Once there is

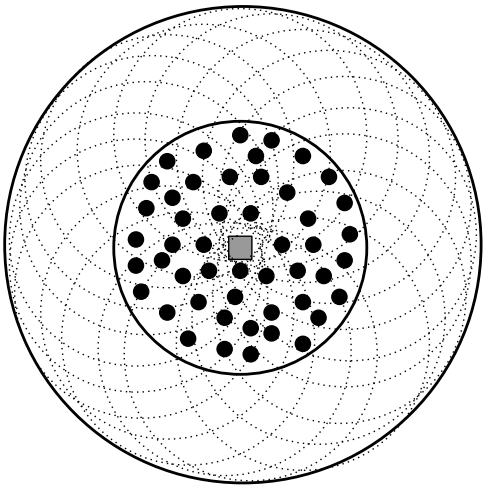


Figure 3.1: If every node has the same fixed transmission radius, then the intersection of all the reachable areas of nodes in the first level is also circular and therefore the network can be naturally thought of as a series of concentric circles.

a circle in the centre of the network the rest of the network naturally follows in concentric circles because the intersection of the reachable areas of the sink's children is approximately circular as illustrated in Fig. 3.1. For every subsequent corona, the intersection of the reachable areas of the nodes within that corona is circular and the intersection can be called the corona's reachable area. Therefore, any sensor network is conceived of as being circular because the reachable area of every corona is circular. Some non-circular networks can also be modelled as circular, for example square networks may be thought of as being circular with added nodes.

3.1.2 Single, Resource-Unconstrained, Central Sink

Every sensor network must contain at least one node capable of processing the gathered data or at least acting as a gateway between the network and either the end user or another network like the Internet. This node is referred to as the data sink (or simply sink) and must be more powerful than the other nodes because it performs far more work. Since the sink is deemed to have so many more resources than the other nodes it is often simpler to talk about the sink as being resource-unconstrained and to imagine that it has infinite resources because it will certainly have enough resources to outlast the rest of the nodes. For clarity,

the term *sensor nodes* is used to refer to all the nodes that are not sinks.

It is sometimes necessary for a network to have more than one sink, for example if the network covers an extremely large physical area which would mean that the scalability of the network is limited by having only one sink. However, it is desirable to minimise the number of sinks because they are considerably more expensive than the sensor nodes. If money were no issue then network performance could be massively improved by replacing every sensor node with a sink; clearly a sensor network is designed to have few sinks.

From an analytical point of view it should be assumed that there is only one sink for three reasons. Firstly, a single sink simplifies the mathematical analysis; secondly, the single sink generally represents the worst case scenario which is the appropriate case to study; and thirdly, even in networks with many sinks, nodes are assumed to communicate only with their closest sink and therefore these networks can be considered as collections of many single-sink networks.

That a many-sinked network can be thought of as many single-sink networks also gives a justification for assuming a central sink. All nodes transmit to their nearest sink and so naturally the sink becomes the centre of the network. This also follows from the discussion above regarding the circular nature of the network, since the network can be thought of as growing in circles around the sink. Finally, the centre has been proven to be the optimal position for the sink in terms of both latency and energy efficiency in multi-hop networks [LH05].

3.1.3 Static Nodes

Whether or not the nodes in a network are mobile is often not a design decision but a function of the application and the target environment. For some applications mobility is desirable, for example a sensor network designed to track the movement of animals may be best served by attaching sensors to the animals themselves. Other applications have inevitable mobility whether desired or not, for example if the network is deployed in moving water. However, in many cases mobility is not needed and serves only to cause problems from changing topologies. In particular, where the phenomena being sensed are relatively static and the target environment is stable, the sensor nodes are likely to be static as well. These applications include the data gathering applications considered in

this thesis, for example the volcano, greenhouse and glacier monitoring applications mentioned in Chapter 1. In these scenarios, modifying the nodes to make them mobile is expensive and the mobility brings significant challenges and performance degradation owing to the frequent changes of the wireless links between nodes as they move.

It is worth noting, however, that a small amount of mobility can occur without necessarily affecting the routing tree of the network. In practical networks there will be fading effects that cause the signal strength between nodes to fluctuate by small amounts in hard to predict ways. To ensure high quality communication some lee-way must be given so that sometimes the signals are more powerful than they need to be but at other times they are just powerful enough. If some extra power is being used in the transmissions as a safeguard then a small amount of mobility may not overly affect the link quality between nodes and would not result in the breaking of wireless links. In effect it would be as if the node did not move at all.

3.1.4 Uniform Random Distribution

In a similar way to the reachable area described above, each node has a *coverage area* surrounding it in which it can detect phenomena. It is obviously desirable to have as large a physical area covered by sensor nodes as possible and this is achieved by spreading the nodes evenly through the network area in a uniform distribution. In practice, however, due to the large number of nodes it is usually not possible to position the nodes exactly as desired and so there will be some randomness in their placement. To model this, the node locations are calculated as a poisson point process, meaning that their positions in the network are determined independently of each other with coordinates drawn from a uniform random number generator. For mathematical analysis the density is assumed to be constant everywhere but in practice there will be some small variations.

There are two widely used algorithms for generating uniformly distributed points inside a circle: rejection sampling and polar coordinates. Rejection sampling imagines a square around the circle where the length of each edge of the square is equal to the diameter of the circle. Points are then randomly generated inside the square, which is straightforward, and each point is tested to determine whether it

is inside the circle or not. Points are rejected if they are outside the circle.

The polar coordinate method uses a uniform random number generator to create a random angle, θ, and a random distance from the centre, r, and from those calculate x and y coordinates of a point inside the circle, following equations (3.1) and (3.2). A slight complication with this method is that when calculating the distance from the centre, the square root of the random number must be used to ensure a uniform distribution. Rejection sampling generates points much faster but, theoretically at least, does not guarantee that it will return; it could loop forever generating points inside the square that are outside the circle.

$$x = r\cos(\theta) \tag{3.1}$$

$$y = r\sin(\theta) \tag{3.2}$$

For the simulations in this report the rejection sampling method is used. Not only is it faster but in Chapter 8 this assumption is relaxed and a scenario with a Gaussian distribution of nodes is considered. It is straightforward to modify the rejection sampling method to accommodate this change.

3.1.5 Homogeneity

Homogeneity or heterogeneity is a design decision and each choice has advantages and disadvantages. Romer and Mattern suggested that nodes are cheaper in homogeneous networks because of economies of scale [RM04] but Mhatre and Rosenberg argued that the opposite is true because in a homogeneous network all nodes must have the capacities needed by the most complex node whereas in heterogeneous networks some nodes can be simplified and hence cheaper [MR04b]. For data gathering networks all sensor nodes must perform the same operations and therefore no node needs to have more or less hardware than any other which suggests that a homogeneous network would be cheaper. Another advantage of using a homogeneous network is that only a single program needs to be designed which can then be loaded onto all sensor nodes.

A consequence of homogeneity is that all nodes have the same batteries and therefore the same initial energy. Although there will always be some variation

in battery capacities due to their manufacturing this is likely to be small and can be ignored.

3.1.6 Connectivity

It is obviously desirable that the network be well connected so that all data reaches the sink and to prevent routing holes. The minimum required node density (measured in terms of the number of neighbours) to ensure connectivity has been well studied. Kleinrock and Silvester famously suggested that six neighbours was the "magic number" [KS78]. This number was then revised upward to eight by Takagi and Kleinrock [TK84] and Stojmenovic and Lin found a similar result for sensor networks [SL01]. Therefore, to ensure that the network is indeed well connected the minimum density that is used in this thesis is ten neighbours per node. However, higher densities are also used for evaluation.

3.1.7 Network Lifetime

Among the assumptions made by Li and Mohapatra was that the nodes generated constant bit rate data. However, in data gathering networks the typical application involves nodes querying their sensors periodically which more naturally leads to discrete rounds each of which is as long as the time between successive sensor readings. It is also assumed that each round is long enough that all the data gathered during it can reach the sink before the next set of sensor readings are taken. This is necessary for networks without data aggregation because if a node cannot empty its buffers from one round of data before it starts receiving the next set of data then it will eventually fill its buffers and start losing data.

3.1.8 Fixed Size Data Packets

Since the network is assumed to be homogeneous with every sensor running the same program and sensing data at the same rate, it is reasonable to assume that the data packets they generate will be the same size.

3.1.9 Multi-Hop Communication

This is a straightforward result of the low power of sensor devices combined with the relatively large size of the network. Since the nodes have limited capabilities they cannot transmit directly to the sink and must rely on nodes that are physically closer to the sink to relay their packets for them.

3.1.10 Network Capacity

In order to ensure the proper operation of the network, each node must have enough capacity to handle all the traffic required of it. This means that it must have enough memory to buffer all incoming packets until it is able to forward them.

3.1.11 No Aggregation

As discussed in the previous chapter, data aggregation is an extremely powerful tool for reducing the energy consumption in a network. The best form of aggregation is full aggregation in which a node is able to compress all incoming data packets into a single outgoing packet of the same size as a single incoming packet. This applies to simple functions such as `MAX`, `MIN` or `SUM`. For many networks, though, these kind of operations are not applicable and the amount of outgoing data increases with an increase in incoming data. If full aggregation is not possible then it may be that partial aggregation can be used. In partial aggregation the number or size of outgoing packets increases as more packets are incoming but some compression or merging of data is possible such that there are fewer outgoing bits than the sum of all incoming bits.

In this thesis, however, the assumption is that no aggregation takes place at all so that every bit that is received is forwarded. This is simpler to model than partial aggregation and, as discussed in Section 2.2.1, full aggregation is not always viable. However, it is worth noting that the theory and approaches described in this thesis apply to networks with partial aggregation as well. So long as aggregation is not perfect, the amount of data that requires forwarding increases closer to the sink leading to the energy hole problem. Maximising inner-corona

balance mitigates this problem.

3.1.12 Ideal MAC Layer

In order to properly analyse the effect of the routing layer it is necessary to isolate its effects from the other layers. The simplest method for this is to assume that the other layers of the protocol stack perform perfectly. This is especially important in the MAC layer which is responsible for making sure that sensors do not interfere with each other. The MAC layer is also responsible for minimising the energy required to transmit and receive packets by timing the transmissions and allowing the radio to be switched off as often as possible. For a survey on MAC algorithms designed specifically for sensor networks see Demirkol *et al.* [DEA06].

3.1.13 Unit Disk Model

The unit disk model states that in wireless communication, the packet reception rate (PRR) between two nodes is binary depending only on the distance between them. That is, if the two nodes are within some defined transmission range then all packets between the two are received with no errors whereas if they are further apart than that range no packets can be sent between them. Stojmenovic *et al.* point out that the expected packet reception rate (PRR) depends on distance and behaves in a very similar way to the UDG model's predictions, as shown in Fig. 3.2 [SNK05]. The UDG model is very widely used because of its simplicity and because it allows for strong mathematical analysis.

This model has been called into question, however, because of the existence of a transitional region in which packet reception rates vary (see Fig. 3.3 below) [ZK04]. A simple justification for its use is to note that the unit disk model is an accurate model of the behaviour in position based routing if nodes are restricted to using relays from within the connected region. However, previous research has shown that it is most energy efficient to use nodes inside the transitional region as relays and to use both distance and packet reception rate together to calculate the cost associated with a given relay [SZHK04].

To the best of my knowledge no attempt has been made to justify the use of the unit disk graph (UDG) model in light of the transitional region. In this chapter

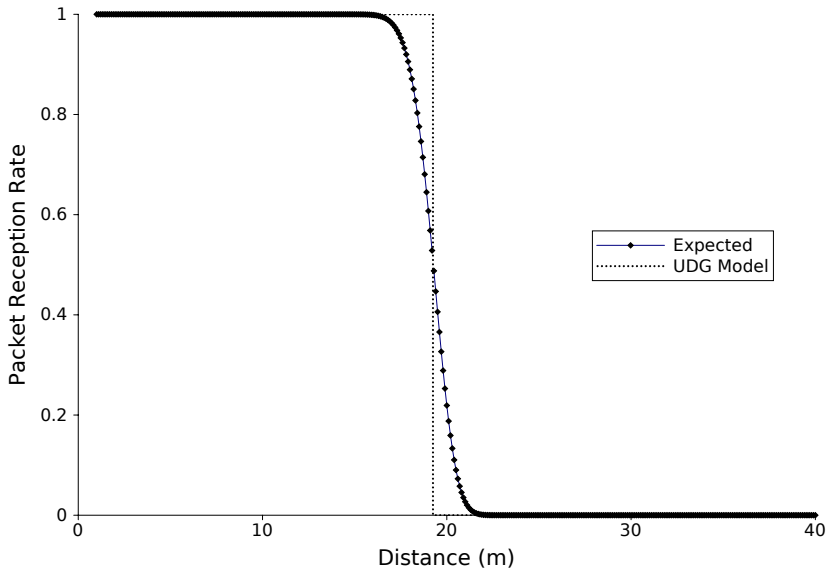


Figure 3.2: The expected packet reception rate depends on distance and the UDG model is a good approximation of the expected behaviour.

an attempt to do so is made by first showing, in the next section, that it is more energy efficient to use a blacklisting strategy for position based routing which would result in the binary nature of links that is used in the UDG model. In the section after that the UDG model is shown to be a close approximation of the performance of the energy efficient blacklisting strategy which further justifies its use.

3.2 Blacklisting for Position Based Routing

3.2.1 Background

Position based routing assumes that all nodes know their location and can share that information. Nodes are then able to use that knowledge to decide who to forward their packets to by incorporating it into a metric or cost for every available link. Initially, because the UDG model was assumed, position based routing took the progress of a link as the metric where progress was a measure of how much closer the packet would get to its final destination by being sent along the link. This metric was theoretically optimal because it minimised the number of relays that the packet needed to be transmitted through. In the UDG

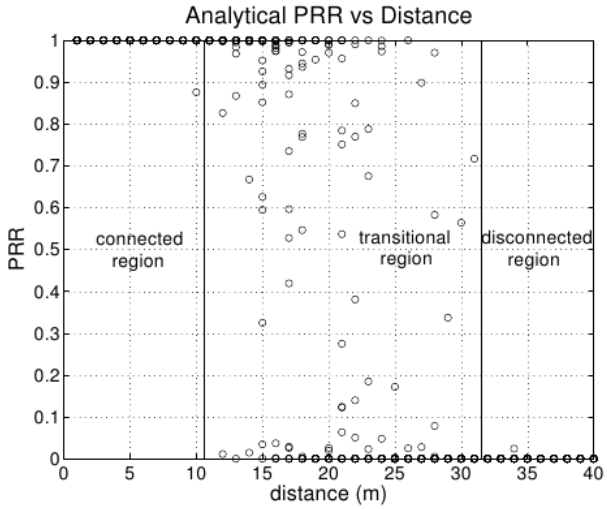


Figure 3.3: In the real world three distinct regions exist around a transmitting node each displaying different behaviours of the packet reception rate (PRR). Image taken from [ZK04].

model all existent links are perfect and so questions of transmission failures and retransmissions do not apply.

However, the characteristics of the wireless channel have been well studied since then and all the studies have shown that the UDG model ignores the potentially large transitional region [GKW+02, ZG03, WTC03, ZHKS04, CABM05]. As Fig. 3.3 illustrates, there are three regions in the wireless channel. The UDG model includes the connected and disconnected region where the packet reception rate (PRR) is either 100% or 0% respectively. However, it ignores the transitional region where the PRR of a link can vary considerably and two links at the same distance may have different PRR. Using links in the transitional region raises the issue of failures and retransmissions.

Zuniga and Krishnamachari proposed a more accurate model for the relationship between PRR and distance [ZK04]. They based their model on the log-normal shadowing model which has been shown to be statistically valid even for low-power devices [SMP99]. The model, given in equation (3.3), predicts the path loss in decibels of a transmitted signal based on the path loss, $PL(d_0)$, at a reference distance, d_0, the path loss exponent η and a zero-mean, Gaussian random variable X with standard deviation σ.

$$PL(d)_{dB} = PL(d_0)_{dB} + 10\eta\frac{d}{d_0} + X_\sigma \tag{3.3}$$

Based on the shadowing model, the signal-to-noise ratio (SNR) at distance d, $\gamma(d)$, is predicted (in decibels) by equation (3.4) where P_t is the transmit power, $PL(d)$ is the path loss at distance d as predicted by the log-normal shadowing model and P_n is the noise floor.

$$\gamma(d)_{dB} = P_{t\ dB} - PL(d)_{dB} - P_{n\ dB} \tag{3.4}$$

For non-coherent frequency shift keying, the PRR model is given in equation (3.5), where b is the number of bits in the packet.

$$PRR(d) = \left(1 - \frac{1}{2}\exp^{-\frac{\gamma(d)}{2}\frac{1}{0.64}}\right)^b \tag{3.5}$$

Based on this model, Seada *et al.* analysed the trade-off between the length and the PRR of links [SZHK04]. On the one hand, choosing long links reduces the number of hops required to reach the destination which can lower the total energy usage. However, long links are more likely to have low PRR and therefore require retransmissions. On the other hand, shorter links, while likely having higher reception rates, require more hops. Therefore, they argued that using only the length or only the PRR of links between nodes as the metric would result in poorer performance than considering both.

In their analysis, Seada *et al.* defined energy efficiency as in equation (3.6) where b_{src} is the number of bits generated by a source, Γ is the proportion of bits sent by the source that are eventually received at the destination, e_b is the energy consumed for each bit transmitted and b_{sent} is the total number of packets sent by the network in order to provide the delivery rate Γ.

$$E_{eff} = \frac{b_{src}\Gamma}{e_b b_{sent}} \tag{3.6}$$

The value of b_{sent} depends on the PRR of the chosen links and the efficiency is analysed for cases with and without automatic repeat request (ARQ). In both

cases they found that the efficiency is maximised by selecting the link with the maximum PRR×distance. They compared their strategy to a number of black-listing alternatives including distance-based and reception-based. In all cases they found that the most energy efficient strategy was to select links that maximised the PRR×distance metric.

This result was generalised by Lee *et al.* who argued that, if the distance between source and destination is relatively large, then the total cost of sending a packet from source to destination is given by equation (3.7) where *Distance* is the total distance between source and destination [LBB05]. In order to minimise the total cost, the value of $\frac{Link\ Cost}{Link\ Length}$ must be minimised which is the equivalent of maximising $\frac{Link\ Length}{Link\ Cost}$. Lee *et al.* termed this last fraction the *normalised advance*. This framework allows the metric used to measure the link cost to be changed as desired while still considering the trade-off between link length and link cost. Stojmenovic *et al.* independently proposed the same framework, calling it *cost per progress* [SO05].

$$\begin{aligned} Total\ Cost &= Link\ Cost \times Hop\ Count \\ &= Link\ Cost \times \left\lceil \frac{Distance}{Link\ Length} \right\rceil \\ &\approx Distance \times \frac{Link\ Cost}{Link\ Length} \end{aligned} \tag{3.7}$$

This framework, and indeed the PRR×distance metric, have been used by some later researchers, e.g. Park *et al.* who added residual energy into the metric [PBC10].

3.2.2 Variable Link Cost

The analysis of Seada *et al.* that found that PRR×distance metric was optimal was performed for two scenarios: a network using ARQ and one not using it. However, in the scenario where the network did make use of ARQ, every link used it regardless of the PRR of that link. In their analysis, ARQ is a function of

the network rather than the individual link. However, I argue in this section that ARQ ought to be a function of the link as well. That is, a network-wide decision must be made as to whether any links can use ARQ to improve their effective quality but if it is decided to use ARQ then its actual use on a given link should depend on the quality of the link. The result is that the energy cost of links in networks that use ARQ is more variable than Seada *et al.* considered.

Suppose that for every link selected for routing, a minimum proportion of packets, q, must be received successfully. If a link quality is too low ($PRR(d) < q$) then ARQ is required for all packets on that link in order to raise the effective reception rate. The purpose of ARQ is to keep the transmitting node informed about the reception of its transmitted packets which allows it to retransmit any that failed to arrive. By using ARQ, a link whose PRR is below the set threshold, q, can nevertheless have the desired proportion of packets received by retransmitting some of those that failed to be received the first time they were sent.

If the threshold value, q, is less than 1, not all failed transmissions need be repeated as some dropped packets are acceptable. Nevertheless, ARQ is still required for all packets on the link because the source node must first know that a transmission failed before it can decide whether a retransmission is necessary. Note, also, that while the receiving node only transmits one acknowledgement per packet irrespective of the number of attempted transmissions, the source node must be in receive mode after every transmission, even unsuccessful ones, in case an acknowledgement is sent and this consumes as much energy as receiving an acknowledgement after every transmission.

In contrast, if the link is acceptable to begin with ($PRR(d) \geq q$), then there is no need to use ARQ for any packet on that link. Even if packets are not received successfully, the rate at which this happens is acceptable (by definition) and so retransmissions are not required for any packet. Having the receiving node acknowledge packets is simply a waste of energy which does not bring any improvements to performance. To avoid wasting energy in unnecessary acknowledgements, ARQ should be considered a function of the link and not just the network.

By considering ARQ a function of the link, the total energy consumed along a link varies considerably with its quality. Let e_{tx} and e_{rx} be the energy cost of transmitting and receiving a data packet respectively. Then let α be the relative

size of an acknowledgement to a data packet. This generalises from previous works which have assumed that acknowledgements are the same size as data packets [SO05, KNS05]. Given this, the total energy cost to the transmitter of transmitting a packet and receiving an acknowledgement is:

$$E_{tx} = e_{tx} + \alpha e_{rx} \tag{3.8}$$

while the cost to the receiver of receiving the packet and transmitting the acknowledgement is:

$$E_{rx} = e_{rx} + \alpha e_{tx} \tag{3.9}$$

Combining the two, the total energy cost of using a perfect link with ARQ is:

$$E_{link} = (1 + \alpha)(e_{tx} + e_{rx}) \tag{3.10}$$

However, if ARQ is only used for links with below threshold PRR then the total energy cost of a link changes and depends on the PRR:

$$E_{link} = \begin{cases} \frac{q}{PRR(d)}(1 + \alpha)(e_{tx} + e_{rx}) & PRR(d) < q \\ e_{tx} + e_{rx} & \text{otherwise} \end{cases} \tag{3.11}$$

It is clear that the cost of a link with an unacceptable PRR might be significantly higher than one with an acceptable PRR and therefore those low quality links should be avoided. In the next section I use this new link cost function to show that blacklisting links based on PRR is more energy efficient than the PRR×distance metric.

3.2.3 Absolute Reception Based Blacklisting

The cost function defined in equation (3.11) does not lend itself to mathematical analysis. However, it can be evaluated through numerical testing.

As discussed above, in 3.2.1, Lee *et al.* proposed the normalised advance framework for finding the optimal links. In this framework, the optimal link is the one

Variable	Value
η	4
σ	4
$PL(d_0)_{dB}$	55dB
P_t	0dBm
P_n	-115dBm
e_{tx}	16.5
e_{rx}	9.6
q	0.99
α	1

Table 3.1: Summary of the model variable values used in the simulations

that maximises the ratio of link progress to link metric. The simplest method of calculating the link progress is as the distance between source and receiver which is simply the link length. Other methods exist such as the difference between transmitter-to-destination distance and the receiver-to-destination distance but for this thesis the link length is taken as the link progress for simplicity. In this section, the normalised advance framework is used, in conjunction with the link cost function derived in the previous section, to show that it is more energy efficient to blacklist low quality links and then select the longest remaining link, than to combine link quality and cost into the single PRR×distance metric. This blacklisting method was earlier analysed by Seada *et al.* using their cost function and they called it absolute reception based blacklisting (ARB).

The first step towards showing that ARB is more energy efficient than PRR×distance is to show the range of values of normalised advance using the new cost function of equation (3.11). The analysis in this section uses the same values as Zuniga and Krishnamachari, namely that the $\eta = 4$, $\sigma = 4$, $PL(d_0)_{dB} = 55$dB, output power is 0dBm and the noise floor is taken as -115dBm. Since the Zuniga model was derived specifically for the Mica2 sensor node, the values of e_{tx} and e_{rx} are taken from the CC1000 which is the radio that the Mica2 node uses. For convenience the current consumption is converted directly into energy consumption noting that the exact values are not important so long as the ratio between them is preserved. Therefore $e_{tx} = 16.5$ and $e_{rx} = 9.6$. The new variables are set as: $q = 0.99$ and $\alpha = 1$ meaning that the acknowledgements are the same size as the data packets. These settings are summarised in Table 3.1.

With these settings, the value of the normalised advance ($\frac{E_{link}}{d}$) was calculated

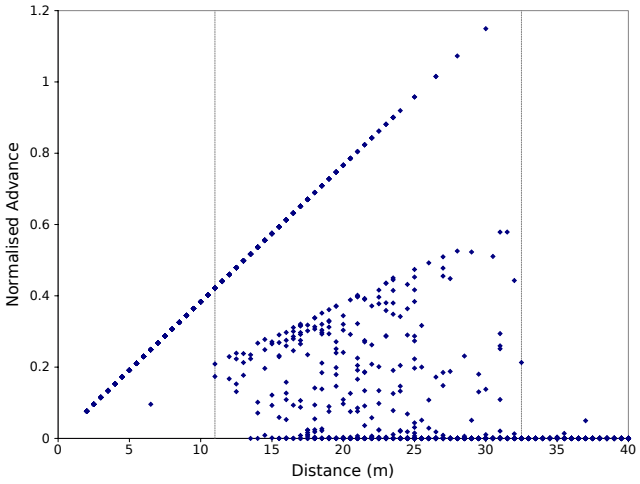


Figure 3.4: The optimal links, as measured using the normalised advance framework, are likely to be in the transitional region. However, links in that region may also be sub-optimal.

over a range of values of d ranging between 2m and 40m at 0.5m steps. For each value of d, 25 values were calculated to allow for the variation of the random variable in the log-normal shadowing model and each value was included in the plot in Fig. 3.4. It can be clearly seen that the optimal links are inside the transitional region. However, links that extend into this region may also have lower metric values and be worse than links in the connected region.

What is striking is that there are two separate patterns. Starting in the connected region and continuing to the end of the transitional region, one set of normalised advance values increases linearly with distance. The second set is found almost exclusively inside the transitional region and shows a variation in values but even its maximum values are significantly lower than those of the first set. The explanation is that the two groups are the result of the two different link costs from equation (3.11). The first set, which starts in the connected region which always has high PRR, includes all links with above threshold PRR values. The second set is the normalised advance values of the links that needed to use ARQ. This shows that the links that maximise normalised advance, which are the optimal ones, are those with high PRR values which suggests that only links with above threshold PRR values should be used for routing.

To clarify the initial results the PRR values of the optimal links were examined. The value of α was varied from 0 to 1 in steps of 0.1 and for each value the link that

α	Mean	Standard Deviation
0.0	0.964	0.061
0.1	0.976	0.05
0.2	0.985	0.041
0.3	0.991	0.03
0.4	0.995	0.021
0.5	0.997	0.016
0.6	0.998	0.009
0.7	0.998	0.007
0.8	0.998	0.006
0.9	0.998	0.003
1.0	0.998	0.003

Table 3.2: The mean and standard deviation for the PRR of the optimal links using the normalised advance metric

maximised normalised advance was found and its PRR recorded. This process was repeated 10,000 times for each α value to obtain a view of the PRR value of the optimal links which is summarised in Table 3.2. The results show that the optimal links are those with high PRR and that the more costly the ARQ packets are, the higher the PRR of the optimal links are ($r = 0.785$, $p = 0.004$)[1].

Using the same experimental setup but varying q, the proportion of optimal links whose PRR was greater than q was also recorded. The results, shown in Fig. 3.5, show that as the cost of the control packets increases, a higher percentage of the optimal links are above the minimum PRR threshold ($0.785 \geq r \geq 0.774$, $p \leq 0.005$). For all values of q the percentage of optimal links with above threshold PRR reached 99.9% when $\alpha = 1$, which is when control packets are the same size as the data packets. Additionally, the lower the threshold q is, the higher the proportion of optimal links have $PRR \geq q$. This is to be expected because if q is lower then there are more links with $PRR \geq q$.

The results suggest that the most energy efficient routing method is to first blacklist the below-threshold links and then from amongst the rest select the one with the most progress. This is in contrast to earlier conclusions by Seada *et al.* that suggested that combining progress and PRR into one metric was optimal. In the next subsection simulation results are used to verify this conclusion by directly comparing the energy efficiency of the PRR×distance metric to the ARB

[1]See Section 3.4.4 for a discussion on these and other statistical measures.

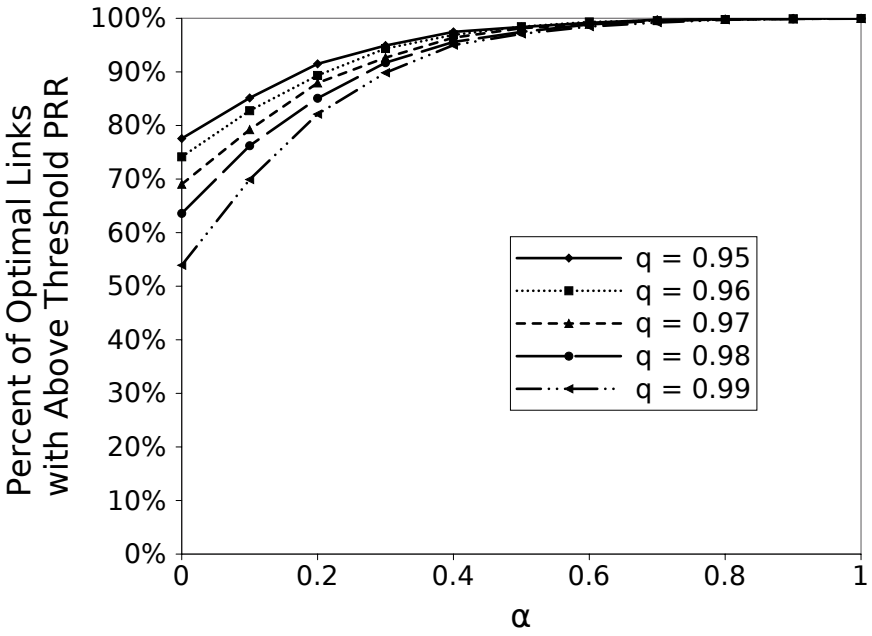


Figure 3.5: The more costly acknowledgements are, the more likely it is that the optimal links will have above threshold PRR values.

strategy.

3.2.4 Simulation Validation

In order to validate the analysis, simulations were conducted comparing the energy consumption of ARB with the PRR×distance metric. A network area of 100m x 100m was simulated in which nodes were randomly distributed. In each simulation, 100 packets were sent from a source to a destination, with a new source and destination randomly selected for each packet. The network parameters are those summarised in Table 3.1 but q is set to 1.0 to ensure delivery of every packet.

Fig. 3.6 shows the average energy consumption along each packet's route for both ARB and PRR×distance for varying network densities. For convenience, density is defined in terms of the distance between neighbouring nodes if they were arranged in a perfect grid. That is, a density of 2m means that the distance between neighbouring points on the grid is 2m which, in a 100m x 100m network, results in 2500 nodes.

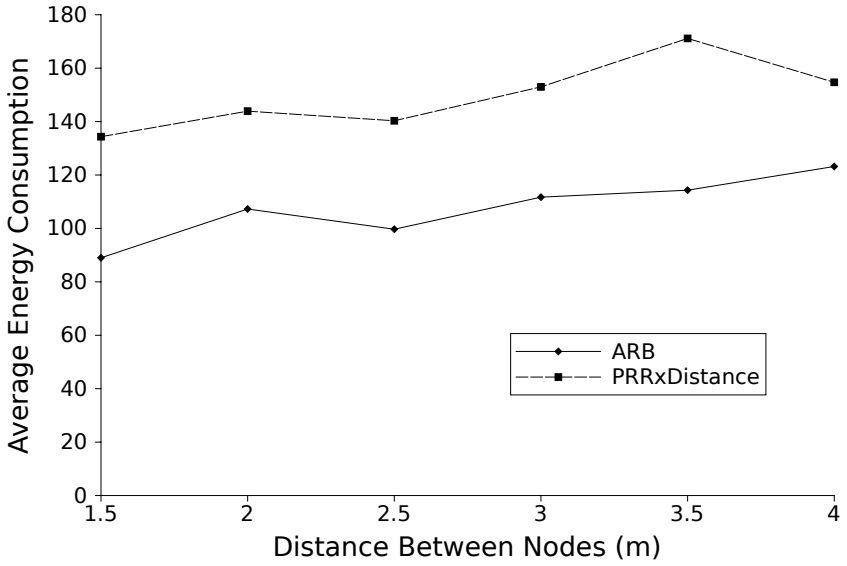


Figure 3.6: The ARB strategy is more energy efficient than the PRR×distance metric, consuming between 26% and 51% less energy.

The results show that ARB consumes less energy than PRR×distance. The improvement is between 26% and 51% though there is no significant correlation between the amount of improvement and density ($p = 0.37$).

In a second series of experiments, the values of α and η were varied. The density was kept constant with an inter-node distance of 2.5m which was chosen because the average observed improvement at that density in the previous experiment (41%) was closest to the average improvement (40%) over all density values. For each combination, the ratio of the average energy consumption under ARB and PRR×distance was calculated. A ratio less than one indicates that the PRR×distance strategy consumes less energy than ARB, whereas a ratio greater than one indicates that ARB is consuming less.

The results shown in Fig. 3.7 show that as the cost of acknowledgements (represented by the value of α) increases, ARB consumes the least energy and becomes increasingly more efficient. This suggests that ARB certainly outperforms PRR×distance when acknowledgements are of a similar size to the data packets and are not significantly worse in other cases. Note, though, that the prevailing assumption is that acknowledgements in sensor networks, particularly data gathering networks, are likely to be a similar size to data packets [SO05, KNS05]. The results are therefore sufficient to strongly suggest that ARB is a more energy

efficient approach to position based routing than PRR×distance.

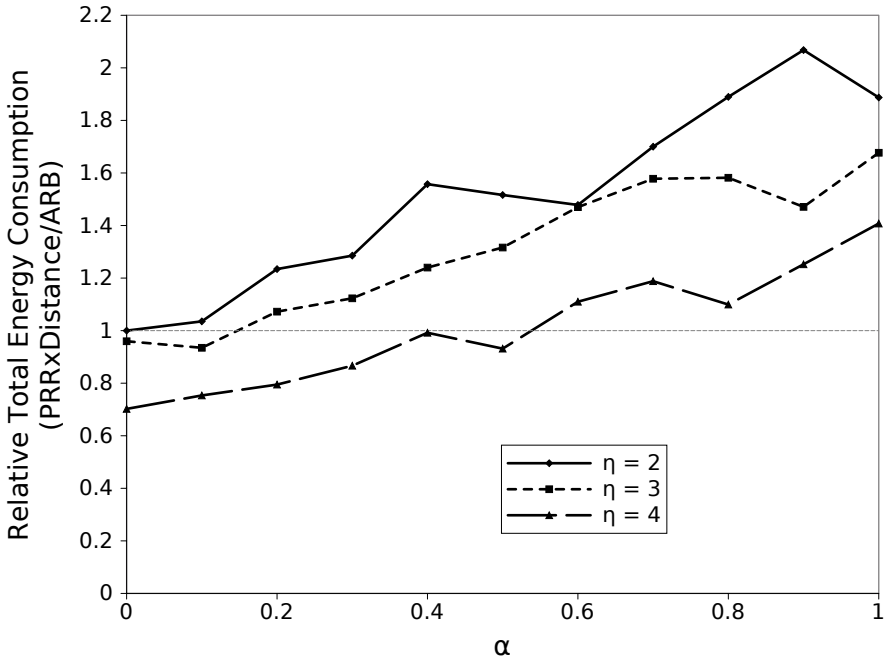


Figure 3.7: The ARB strategy is generally more energy efficient than the PRR×distance metric except when both the path losses are high and the acknowledgements are much smaller than the data packet.

3.3 The UDG Model as an Approximation of ARB

The results from the previous section show that for energy-efficient position-based routing, the optimal links are almost exclusively those with above-threshold PRR, at least in cases where the control packets are of similar size to the data ones. This fact means that ARB is more energy efficient than PRR×distance but also leads to a strong justification of the UDG model. Given that the optimal links are almost always ones with high enough PRR it is reasonable to argue that from the perspective of the routing protocol there are only two types of links: links with above threshold PRR that are deemed acceptable and may be considered for routing and all other links which are unacceptable and should be ignored. The acceptable links have effectively perfect reception rates since no ARQ or retransmissions are ever required on them. On the other hand the unacceptable

links may as well have 0% reception rates since they are excluded from the routing algorithm's consideration. This is exactly the binary link status assumption of the UDG model and so the results from the previous section justify one key element of the model.

However, there is another element of the UDG model which still requires justification, namely the assumption that the cross-over point between perfect links and others is at some fixed distance. This assumption is plainly a simplification of the reality and cannot be true for individual links. Nevertheless, in this section I will argue and then show that the standard UDG model with a fixed transmission radius will serve as a good approximation to the performance of the ARB strategy which would then justify the use of UDG. Caution would still be needed because although it would be appropriate to use UDG in simulations because of its simplicity, the actual routing strategy used in the real network would have to be ARB. That is, the links that a node considers for routing must be determined by ARB rather than UDG in any real network.

The reasoning behind the expectation that UDG would approximate ARB starts from a return to the initial justification for UDG given by Stojmenovic *et al.* mentioned above in Section 3.1.13, namely that the expected packet reception rate is dependent on distance only and is closely matched by UDG. Under the ARB strategy, the chosen links must all have a similar PRR value (assuming that the value of q is relatively high) because they must have a PRR at least equal to q. The Gaussian relationship between link PRR and link length means that the links with a given PRR will be normally distributed around a certain length. With enough links the average link length would converge to a fixed value which could then be taken as the transmission radius of the UDG model.

As an initial test of this reasoning, Monte Carlo simulations of a simple chain topology were conducted in line with the approach of Seada *et al.* [SZHK04]. The source and destination nodes were placed 1,000m apart with nodes evenly spaced between them. The parameter values were as summarised in Table 3.1 with the exception that $q = 1.0$ to guarantee packet delivery. The distance between the nodes was varied to examine the effect of density and the average energy consumption of 50 runs was recorded. In the case of the UDG model, the transmission radius was tuned in order to match, as closely as possible, the average hop length of ARB.

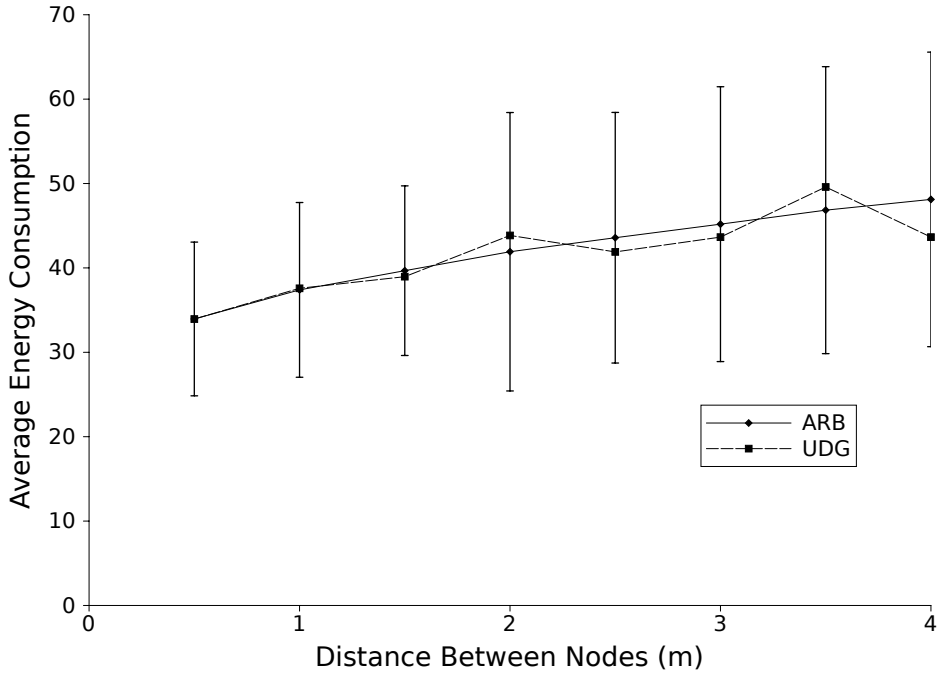


Figure 3.8: The UDG model applied to a simple chain topology is a close approximation to the optimal ARB strategy, although at low densities the two become less similar.

Fig. 3.8 shows the results which confirm that UDG is a close approximation of the ARB strategy. The difference between the two is very low at high densities (<5%) but is higher at low densities (up to 9% difference when the distance between nodes is 4m). The divergence between the two is very strongly related to the density, as indicated by the distance between nodes ($r = 0.90$, $p = 0.002$). At low densities the two methods produce virtually identical results but at higher densities the two diverge although this is perhaps related to the increase in uncertainty in the ARB results. As can be seen from the graph, the confidence interval values increase with an increase in inter-node distance ($r = 0.905$, $p = 0.002$) and the UDG average falls well within the interval.

A further set of simulations, similar to those carried out for ARB described above, consider a network of nodes randomly and uniformly deployed. The method is as described previously and the results are shown in Fig. 3.9. The results are similar to those found in the chain topology. The UDG model is a close approximation of the ARB strategy with the difference between them being no more than 9% which falls within the 95% confidence interval of the performance of ARB. Indeed, over the range of densities examined, there is no statistically significant difference in

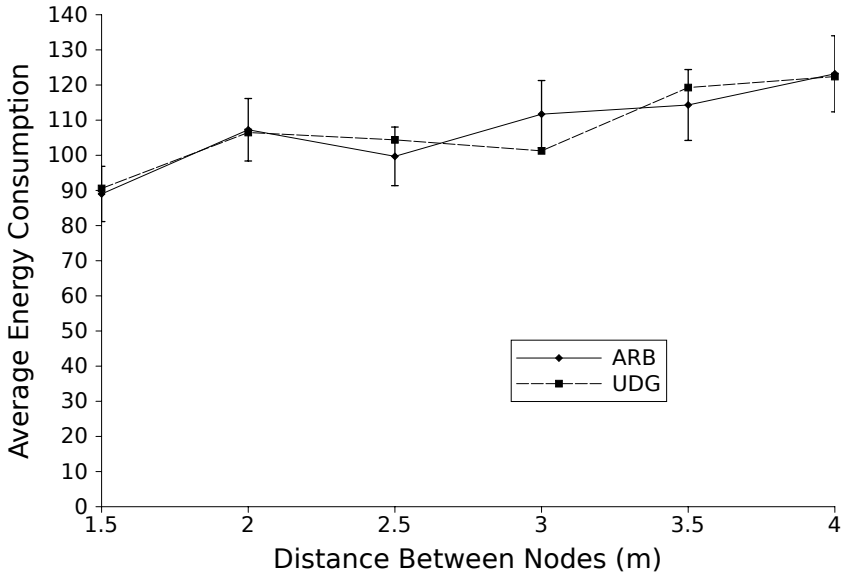


Figure 3.9: As with the chain simulations, the UDG applied to a network is a close approximation to the ARB strategy.

the performance of the two approaches ($p \geq 0.135$). Again the variation in the ARB results correlate to the density ($r = 0.945$,$p = 0.004$).

These results show that for every network with a relatively high required threshold, a UDG model can be found that is a close approximation of the performance of the energy-efficient ARB strategy. Although the UDG model is not an accurate reflection of reality and the transmission radius used by it cannot be used in reality this is not significant in simulations. This is because the chosen transmission distances in simulations are somewhat arbitrary in as much as they rely on a large number of factors and usually can be increased or decreased without having any affect on protocol performance. Therefore, if every UDG model is a close approximation of a network using the ARB strategy then protocol behaviour can be justifiably examined in simulations using the simple UDG model. This is a highly significant finding because the UDG model remains widely used.

3.4 Metrics

The primary measures of network performance used in this thesis are the network lifetime and the balance of the final routing tree which, to some extent, is a proxy

for network lifetime. However, there is also interest in measuring the network's connectivity and latency. In this section the metrics used for each of these is described. At the end of the section there is also a brief note about the statistical measures used in this thesis.

3.4.1 Lifetime

Network lifetime is usually defined as the time until the first node in the network depletes its batteries (normally referred to as "dying"). This represents a lower bound on the lifetime and, when the first node dies, the network performance starts to degrade. This is especially true in the networks considered in this thesis where the first node to die is always in the inner-most corona, as proved in Section 1.1, and its death results in the cutting off of part of the network from the sink.

Measuring network lifetime directly can often be unhelpful because there are many factors that affect it which are somewhat arbitrary. For example, the amount of energy that each node starts with will have a direct impact on the lifetime of the network; doubling the initial energy of the nodes doubles the lifetime of the network without changing anything else. Similarly, the power consumed when transmitting and receiving packets and the size of the packets also affect the lifetime and are also arbitrary to some extent. Comparing lifetimes directly may not reveal information about the way that the routing protocol affects lifetime.

In this thesis the aim is to propose new protocols that can create static routing trees which maximise inner-corona balance and maximising the balance maximises network lifetime. It is therefore obviously necessary to have a measure of balance to compare different routing trees directly. Hsiao *et al.* proposed using Jain's fairness index to measure balance [HHKV01]. The index was originally proposed by Jain *et al.* for measuring the fairness of resource allocation in shared systems, assuming that ultimate fairness is for all systems to be assigned the same amount of resources [JCH84]. The index, shown in equation (3.12) where w_i is the allocation to user i and n is the number of users, has been used in numerous other applications. In the case of network balance, Hsiao *et al.* defined the term *top subtree* to refer to any subtree whose root was in level one of the

routing tree and adapted the fairness index so that w_i is the weight of top subtree i. With the assumptions used in this thesis that every node generates the same amount of data and there is no aggregation, the weight of a top subtree is directly proportional to the number of nodes in it, i.e. the number of descendants of the level one node that is its root.

$$\theta = \frac{\left(\sum_{i=1}^{n} w_i\right)^2}{n \sum_{i=1}^{n} w_i^2} \tag{3.12}$$

Jain's fairness index was proposed because it has four desirable properties. Firstly, the result is independent of the population size, that is the number of top subtrees in this case. If an allocation is perfectly balanced, then adding more subtrees with the same weight as the original subtrees should not change the index value. Secondly, the index is independent of the way in which weights are measured so that only the relative values of the weights should be important. An example given by Jain *et al.* is that if the fairness of allocation of incomes is being balanced then the metric should be the same regardless of whether incomes are considered in pounds or pence. Thirdly, the index is bounded between zero and one where one indicates perfect balance which makes it easier to compare different allocations and decide whether a given allocation is well balanced. Finally, the index is continuous and the result changes if the allocation of any one user changes.

The continuous nature of the index makes it usable in greedy algorithms because any change in the allocation is reflected in the index and can be used to judge whether the allocation is more or less balanced as a result of the change. This property also makes it a useful metric for comparing different allocations and the balance index is therefore used in this thesis.

However, the index is not a strong predictor of network lifetime because the lifetime is defined as the time until the first node dies which is determined by the most heavily loaded subtree only. If, for example, the second-most loaded subtree has some of its nodes moved to the lightest loaded subtree then the balance index will increase even though the network lifetime is not changed at all.

In this thesis the max/mean ratio is taken as a proxy for network lifetime so that the impact of the routing tree's balance on lifetime is isolated away from the effects of the radio power consumption rates and the initial energy of the nodes.

The ratio compares the workload of the heaviest loaded subtree to the average workload of all subtrees. In a perfectly balanced routing tree every subtree has the same workload equal to the mean workload and the network lifetime is maximised. In less balanced trees, at least one subtree has a heavier load than the mean load which reduces the network lifetime and increases the max/mean ratio. The precise amount of reduction in lifetime depends on the proportion of extra load on the heaviest subtree and this is, of course, reflected in the max/mean ratio as well. Assuming that after the tree is constructed the major energy consumers are all related to the communication of data packets to the sink, then, for example, if the most loaded subtree has twice the load of the mean the network lifetime is halved compared to the maximum potential lifetime.

The max/mean ratio can also be used to compare the lifetime of two routing trees. Consider two trees, X and Y constructed for the same network with corresponding max/mean ratios of mm_X and mm_Y where $mm_X > mm_Y$ meaning that tree Y is more balanced than X and therefore has a longer lifetime. Let Lf_O be the optimal lifetime of the network and Lf_X and Lf_Y be the lifetime of the network with routing tree X and Y respectively. The lifetime of the network with each tree is reduced from the optimal lifetime in proportion to the amount of load that the heaviest tree has above the mean, which is measured by the max/mean ratio. Therefore, Lf_X can be expressed in terms of the Lf_O and mm_X as follows:

$$Lf_X = \frac{Lf_O}{mm_X} \tag{3.13}$$

With this, the relative improvement, I, in terms of lifetime gained by moving from one routing tree, for example tree X, to another, for example, Y, can be expressed in terms of the max/mean ratios of the trees:

$$\begin{aligned} I &= \frac{Lf_Y - Lf_X}{Lf_X} \\ &= \left(\frac{Lf_O}{mm_Y} - \frac{Lf_O}{mm_X}\right)\frac{mm_X}{Lf_O} \\ &= \left(\frac{Lf_O mm_X}{Lf_O mm_Y} - \frac{Lf_O mm_X}{Lf_O mm_X}\right) \\ &= \left(\frac{mm_X}{mm_Y} - 1\right) \end{aligned} \tag{3.14}$$

3.4.2 Connectivity

One of the trade-offs for lifetime that will be discussed in later chapters is network connectivity which measures the proportion of sensor nodes that are connected to the routing tree. This is a simple percentage calculated as the number of nodes connected to the routing tree divided by the total number of nodes deployed in the network. The sink node is not included in this calculation since it obviously will always be connected to itself.

3.4.3 Latency

A second trade-off is lifetime for latency which is a measure of how long data takes to reach the sink. In this thesis the network lifetime is divided into rounds that are long enough for all data packets to reach the sink which precludes the possibility of measuring latency in time units. Moreover, the actual time taken for a packet to reach the sink depends on numerous factors other than the routing tree, such as the data rate, and it is therefore preferable to provide a measure of balance that isolates the effects of the routing tree. To achieve this, latency is measured as the average number of hops between the nodes and the sink in a given routing tree.

3.4.4 Statistical Measures

There are two important statistical measures used throughout this thesis: the confidence interval and Pearson's correlation coefficient. The confidence interval provides a range of values that will contain the true value of a parameter with a given probability. For example, if many simulations are run for a given network configuration and the lifetime is recorded for each run then this set of data can be used to generate a mean lifetime for that configuration. However, this is no guarantee that the true lifetime is equal to the mean. The 95% confidence interval for the set gives a range of values that has a 95% chance of containing the true lifetime. This is a measure of how good the mean is at approximating the lifetime because if the range is very high then there is significant uncertainty about the true value of the lifetime. In this thesis, the 95% confidence interval is always used and the range is reported as $(\pm CI)$.

The confidence interval is calculated from the set of samples as follows. First the degrees of freedom is found which is simply the number of samples less one. This is taken together with $\frac{1-CI}{2}$ to look up the corresponding entry in the t-distribution table which is widely available. For the 95% confidence interval the relevant index in the table is $\frac{1-0.95}{2} = 0.025$ and the entry with 24 degrees of freedom (ie 25 samples) is 2.064. Finally, the standard deviation of the sample is divided by the square root of the number of samples and the result is multiplied by the value from the table to give the confidence interval value. The range is then the sample mean $\pm$ the confidence interval value.

The Pearson correlation coefficient is a measure of the linear correlation between two variables and has a value in the range $[-1, 1]$. The higher the absolute value of the coefficient, the stronger the correlation, and as a guideline a coefficient with magnitude greater than 0.5 should be considered to represent a strong relationship, while anything lower than 0.3 should be taken as a weak relationship. The coefficient is reported in this thesis as the value of r.

The strength of the relationship is only part of the story though because it is possible for two variables to show a strong relationship in the samples taken but to really not have any relationship at all. Along with the Pearson coefficient, then, is the significance of the relationship which is a measure of the probability that the samples have the observed relationship when the variables themselves have no relationship. The possibility of there being no relationship is termed the *null hypothesis* and the purpose of the significance measure is to provide an indication of how safe it is to reject the null hypothesis. In this thesis the significance is reported as a p value and it is assumed that when $p < 0.05$ it is is reasonable to reject the null hypothesis and accept the observed correlation coefficient as a true representation of the relationship between the variables.

Throughout the thesis, the correlation coefficient and significance are reported in the format ($r = \ldots$, $p = \ldots$), usually following the reporting of the type of the relationship. It is important to note that while the Pearson coefficient only measures linear relationships, it can also be used to measure logarithmic ones by converting one or both of the data sets into logarithmic form. The coefficient and significance were calculated using an online tool created by Wessa [Wes12].

3.5 Simulation Environment

There are a number of simulators available for sensor networks including ns2 [MF], ns3 [ns3] and Castalia [Cas]. However, these are general purpose simulators designed to allow the design of all layers and which measure performance in the traditional manner by simulating the operation of the network throughout its lifetime. As a result it takes significant time to prepare a new protocol for simulation and anywhere from minutes to hours to simulate a given network.

In this thesis the focus is on constructing balanced routing trees and measuring their properties. As discussed in Section 3.4.1, lifetime in this thesis is measured through the balance and max/mean ratio which do not require the passing of packets through the network once the tree has been constructed. Moreover, the work focuses entirely on the routing layer and ideally this layer should be entirely isolated from the other layers of the network. It is apparent that the additional complexity introduced by the existing network simulators is not required.

Therefore, a purpose built simulator was designed which was capable of very quickly generating a routing tree according to some programmed protocol and then taking the measurements of that tree. The resulting simulator was able to test the performance of tree building protocols on even very large networks in a matter of seconds or a few minutes.

The correctness of the simulator was verified by comparing the statistics it gathered when using a shortest path routing tree (see Section 5.3.1) to those found when using the same method in the Castalia simulator. The results verified that the trees produced by both simulators were of the same type (obviously some differences existed owing to random numbers).

Although the simulator used in this thesis allowed very fast prototyping and simulations, it is limited to protocols that construct static routing trees. Its speed of operation derives mainly from the fact that it can construct a tree, measure and finish whereas other simulators would then allow the network to operate until energy depletion. The result is that dynamic protocols cannot be tested since the results for the performance of such protocols requires the traditional method of simulating the running of a network.

3.6 Chapter Summary

This chapter laid the foundations for the rest of the thesis by enumerating and justifying the assumptions that underpin the rest of the work. Although the assumptions I have used are common in the field they are often not fully listed or justified and therefore it was important to do so. In particular, the use of the UDG model in this thesis required strong justification in light of its well known inaccuracies and this was done. To the best of my knowledge, such a strong justification has never been provided before and this is an important contribution to the research community which continues to use the model.

The justification rested on a reassessment of previous results from Seada *et al.* [SZHK04] who had concluded that the optimal method for selecting links was to measure the PRR and progress of each link and select the link which maximised the product of those measures. However, by considering ARQ as a function of the link quality as well as the network it was shown that a blacklisting strategy, ARB, performs better. The ARB strategy for selecting links shares a crucial property with UDG in that both methods reduce all links to one of two types which can be labelled as either perfect or non-existent. This similarity means that UDG is a good approximation to ARB which strongly justifies the use of UDG in simulations for comparing the performance of protocols.

The metrics by which different routing protocols will be compared were also introduced and discussed along with a brief description of the two major statistical measures used in this thesis.

Some of the work presented in Sections 3.2 and 3.3 of this chapter was published in [KF12b].

Chapter 4

Relay Hole Problem

This thesis is based on the widely used corona model of a sensor network, illustrated in Fig. 4.1. The key element of this model is that the circular network area is divided into a series of concentric coronas of fixed width where the width corresponds to the transmission range of the nodes. The assumption is that a node in one corona can use a node in the next corona as a relay. Thus, a node in corona five, for example, uses a node in corona four as its relay and is consequently five hops from the sink. As Olariu and Stojmenovic stated [OS06]:

> "Importantly, the massive deployment of sensors, combined with the fact that the width of each corona does not exceed the maximum transmission range t_x, guarantees communication between sensors in adjacent coronas."

This assumption is examined in this chapter and it is shown that, even in very dense networks, there will be a significant number of nodes that cannot communicate with nodes in the adjacent corona. The inability of a node in one corona to forward its packets into the next (inward) corona is referred to in this thesis as the *relay hole problem*.

Wu *et al.* noted the relay hole problem during their proposal of a non-uniform node distribution strategy to solve the energy hole problem [WCD08]. They mentioned the need to carefully deploy the nodes such that each node has a certain number of relays in the next inward corona to choose from. Using the UDG model they termed the circular area in which a node's transmission can be received as its *reachable area* and stated:

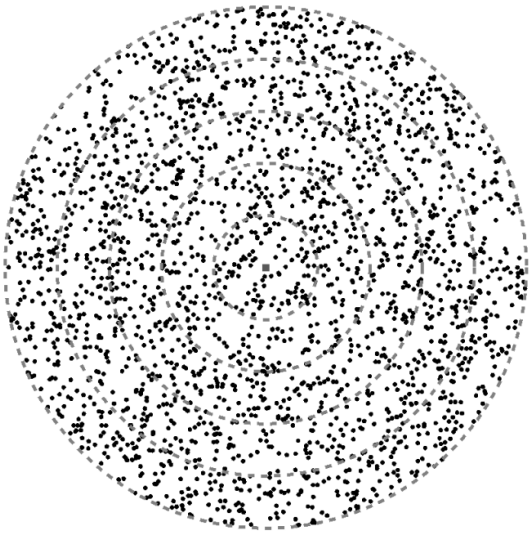


Figure 4.1: A sensor network can be viewed as a series of concentric coronas. The square in the centre is the sink. A node uses intermediate nodes to relay packets to the sink. The relay hole problem causes some packets to pass through more than one node in a single corona.

> "We do not assign nodes on the border of any corona, due to the fact that when a node is placed there, the reachable area in the adjacent coronas reduces to a point."

Thus Wu *et al.* noted the extreme form of the relay hole problem and proposed a solution through careful control of the deployment and positioning of nodes. This method of deployment is obviously considerably more difficult and expensive than a simple random uniform distribution. In many cases it may even be infeasible, especially if the network environment is hostile.

In this chapter, the relay hole problem is analysed from first principles and it is shown that the problem exists not just when nodes are placed on the border of a corona and that the number of nodes affected by it is significant.

4.1 Analysis of The Relay Hole Problem

Under the corona model, the assumption is that all nodes in corona c_i forward their packets to a node in corona c_{i-1} which can be termed the *relay corona*. The model conveniently ensures that any node in corona c_i is i hops away from the

sink and therefore would be in level l_i in a minimum-depth routing tree.

This assumption is based on the fact that the width of each corona is equal to the maximum transmission range of the nodes and thus a node inside one corona can transmit into the next. The area around a node that it can transmit to is known as the *reachable area* (not to be confused with a node's *coverage area* which refers to its sensors). For convenience let the intersection between a node's reachable area and its relay corona be called its *relay area*.

The relay area is the part of the network that contains a node's potential parents. According to the corona model there will always be at least one relay in a node's relay area. The relay hole problem occurs when this is not true and a node's relay area is empty as illustrated in Fig. 4.2. The problem can be viewed as a variant of the routing hole problem which is also when an area around a node is empty. However, in the routing hole problem a node has no neighbours that are closer to the sink than itself meaning that it cannot forward its packet closer to the sink and must re-route effectively behind itself. In the relay hole problem there is at least one neighbour closer to the sink than the node itself but the problem is that that node is not very much closer.

If a node in corona c_i cannot forward its packet to a node in corona c_{i-1}, then it must use another node in the same corona as itself to forward the packet towards the sink. The total number of hops that packets flowing through the node need to traverse in order to reach the sink is increased relative to the optimal if all nodes were positioned ideally.

With the corona model (and the UDG model it relies on), a node's relay area is the area of intersection of two circles, A_{cc}, the formula for which is given in equation (4.1) where R_1 and R_2 are the radius of the two circles and D is the distance between their centre points. The two circles in this case are the reachable area and the circle centred at the sink whose outer edge is the outer edge of the relay corona. If a node in corona c_i is a distance g from the outer edge of that corona, then the radius of its reachable area is simply the transmission range d, the radius of the second circle is $id - d$ and the distance between the centre points of the circles is the distance between the node and the sink which is $id - g$. Inserting these values into equation (4.1) gives the relay area, A_{gi}, of the node as shown in equation (4.2).

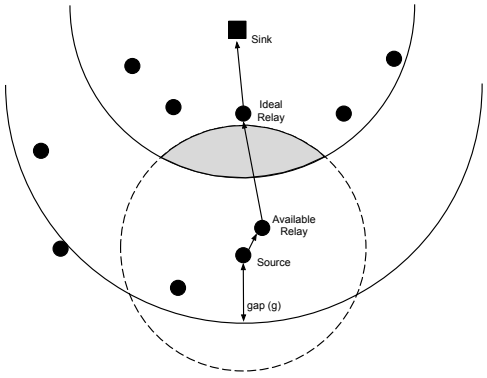


Figure 4.2: The dashed circle is the reachable area of the source node. If its relay area (the shaded area) does not contain any nodes capable of acting as a relay then it must forward its packets around the "hole" using another node in the same corona as itself. This is the relay hole problem which increases latency and reduces energy efficiency.

$$
\begin{aligned}
A_{cc} = {} & R_1^2 \cos^{-1}\left(\frac{D^2+R_1^2-R_2^2}{2DR_1}\right) \\
& + R_2^2 \cos^{-1}\left(\frac{D^2+R_2^2-R_1^2}{2DR_2}\right) \\
& - \frac{1}{2}\sqrt{(-D+R_1+R_2)(D+R_1-R_2)(D-R_1+R_2)(D+R_1+R_2)}
\end{aligned}
\tag{4.1}
$$

$$
\begin{aligned}
A_{gi} = {} & d^2 \cos^{-1}\frac{(id-g)^2+d^2-(id-d)^2}{2(id-g)d} \\
& + (id-d)^2 \cos^{-1}\frac{(id-g)^2+(id-d)^2-d^2}{2(id-g)(id-d)} \\
& - \frac{1}{2}\sqrt{-g^2(2id-2d-g)(id-g)}
\end{aligned}
\tag{4.2}
$$

The probability that a node has the relay hole problem is the probability that every node in the relay corona is outside the relay area. Since the nodes are uniformly distributed the probability that a node is inside a specific sub-area of the network is equal to the proportion of the total network area represented by

the sub-area. The same logic applies to sub-areas of a corona and therefore the probability that a node in the relay corona is outside the relay area is equal to the proportion of the relay corona that it outside the relay area. The probability that every node in the relay corona is outside the relay area is the product of the probabilities for each node in the relay area. These probabilities obviously vary with the corona in question because that affects the area of the relay corona, A_{i-1} and relay area, A_{gi}, as well as the number of nodes in the relay corona, N_{i-1}. They also depend on the gap between the node and the outer edge of its corona, g, which affects the size of the relay area. Thus, for a node in corona c_i which is a gap g from the outer edge of its corona, the probability that it has the relay hole problem, $P(P_i|g)$, is given by equation (4.3).

$$P(P_i|g) = \left(\frac{A_{i-1} - A_{gi}}{A_{i-1}}\right)^{N_{i-1}} \tag{4.3}$$

Since the probability of a node having the relay hole problem depends on the gap between the node and the outer edge of its corona, the probability that a node in a given corona has a given gap must be found as well. Again, since the nodes are uniformly distributed in the network, the probability that a node has a given gap is proportional to the area of the corona taken up by nodes at that gap. Unfortunately, in the corona model the nodes do not take up any space and therefore there is only a line at the specific gap without an area. To solve this problem an annulus is considered with a very small but non-zero width, δg, which is centred at the specified gap. The probability of a node in corona c_i being a gap g from the outer edge, $P(g|i)$ is now calculated as the proportion of the area of the annulus at g, A_{aig} to the total area of the relay corona, A_i:

$$P(g|i) = \frac{A_{aig}}{A_i} \tag{4.4}$$

According to the Law of Total Probability, given in equation (4.5), the probability of a node inside corona c_i having the relay hole problem is given in equation (4.6), where G is the number of annuli and equal to $\frac{d}{\delta g}$.

$$P(A) = \sum_e P(A|B_e)P(B_e) \tag{4.5}$$

$$P(P_i) = \sum_{g=0}^{G} P(P_i|g)P(g|i) \tag{4.6}$$

If $P(P_i)$ is the probability that a single node in corona c_i has the relay hole problem then the number of nodes in each corona with the problem is equal to the probability times the number of nodes in that corona. Therefore, the total number of nodes with the relay hole problem, T, is the sum of this product over all k coronas:

$$T = \sum_{i=0}^{k} N_i P(P_i) \tag{4.7}$$

Equation (4.7) gives only the number of nodes who have the relay hole problem directly. However, there are also indirect effects to consider which come about when another node on the path to the sink has the relay hole problem. The analysis in this chapter is restricted to the case where the first relay node, ie the source node's parent, has the problem and includes the two cases where the source does not and does have the problem.

In the first of these two scenarios, illustrated in Fig. 4.3, the source node has at least one neighbour inside its relay area but every node in that area has the relay hole problem. In this case, termed here as the first secondary effect, the source node becomes an extra hop away from the sink compared to the ideal situation. The probability that this happens, $P(S1_i)$ is the probability that the source node does not have the problem and every node in its relay area does. The number of nodes in the relay area of a node in corona c_i is equal to the proportion of the next inward corona covered by the node's relay area multiplied by the number of nodes in the next inward corona because the nodes are uniformly distributed. Therefore, the probability that a node has the relay hole problem indirectly through the first secondary effect is given by the formula in equation (4.8).

$$P(S1_i) = (1-P(P_i)) \left(P(P_{i-1})^{\frac{A_{gi}}{A_{i-1}} N_{i-1}} \right) \tag{4.8}$$

The second secondary effect, illustrated in Fig. 4.4, is when the source node has the relay hole problem directly and all the nodes it can use as a relay also have

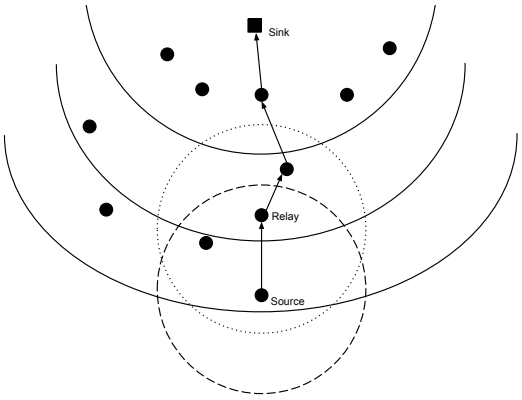


Figure 4.3: The first scenario of indirect effects considered in this analysis is the case where the source node is unaffected directly by the relay hole problem but all the nodes in its relay area are directly affected which has a knock-on effect on the source itself.

the problem, either directly or through the first secondary effect. The second secondary effect results in the source node being two hops further from the sink than it would be in the ideal situation. The probability of this happening, $P(S2_i)$, is the probability that the source node directly has the problem, $P(P_i)$, and every node it can use as a relay has the problem either directly, $P(P_i)$, or through the first secondary effect, $P(S1_i)$. The number of nodes that it can use as its relay can be derived from the uniform distribution of the nodes as the number of nodes in the corona c_i multiplied by the proportion of the node's reachable area that is inside its corona to the area of the corona. Let A_r be the area of intersection between the source node's reachable area and the area of its own corona, A_i, and the probability of the second secondary effect is given by equation (4.9). A_r can be found by subtracting from the reachable area the relay area and the area of intersection between the reachable area and the next outward corona.

$$P(S2_i) = P(P_i)\left([P(P_i) + P(S1_i)]^{\frac{A_r}{A_i}N_i}\right) \tag{4.9}$$

Only the first secondary effect increases the number of nodes with the problem because any node affected by the second secondary effect already has the problem. However, the second secondary increases the impact of the relay hole problem by increasing the added latency even further. Including the first secondary effect, the total number of nodes affected by the relay hole problem, T, must be updated, as in equation (4.10).

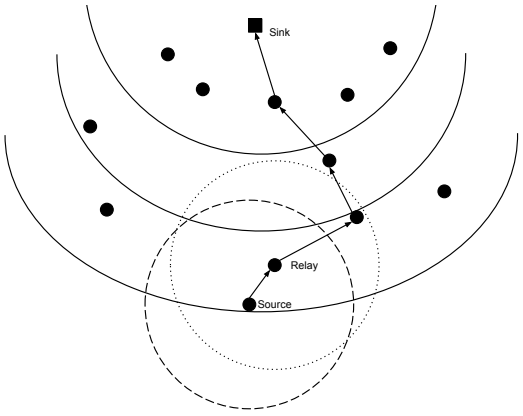


Figure 4.4: The second scenario of indirect effects considered in this analysis is the case where the source node has the relay hole problem and all the nodes it could use as a relay are also affected, either directly or indirectly by it.

$$T = \sum_{i=0}^{k} N_i[P(P_i) + P(S1_i)] \tag{4.10}$$

4.1.1 Key Characteristics

Equation (4.2) shows that the relay area shrinks as a node moves further towards the outer edge of its corona. In fact $\lim_{g\to 0} A_{gi} = 0$. Unfortunately, from equation (4.4), nodes are more likely to be positioned closer to the outer edge of their coronas than the inner edges. The circular nature of the network makes it more likely that a significant number of nodes have the relay hole problem.

It is also apparent from equation (4.3) that the probability of a node having the relay hole problem is lower in coronas further away from the sink. This is because there are more nodes in coronas further away. Even though the relay area for nodes in the outer coronas represents a smaller proportion of the corona's total area, which would tend to reduce the chance of nodes being inside it, the increase in the number of nodes has a greater impact. This implies that as the network radius increases the proportion of nodes in it that are affected by the problem decreases.

Another implication of the relationship between the number of nodes and the probability of a node having the problem, is that the relay hole problem can be mitigated through increased density. By adding more nodes to the network the

probability of a node having the problem decreases. However, from equations (4.7) and (4.10) it can be seen that adding more nodes may increase the total number of nodes with the problem. This is because the probability of an individual node being affected falls but the number of nodes that might be affected increases faster. Nevertheless, the overall proportion of nodes with the problem falls.

4.2 Simulation Validation

The above analysis has been validated through extensive simulations. A circular network is assumed with a single, central sink. Every node has a transmission range of 10m, an initial energy of 10J and generates one 400 bit packet each round which is forwarded to the sink through multi-hop communication.

The energy model is the one first proposed in [HCB02]. The energy consumed transmitting each bit, e_{tx}, is $E_{elec}+\epsilon d^{\eta}$, where $E_{elec} = 50nJ/bit$, $\epsilon = 100pJ/bit/m^4$, d is the transmission distance and η is the path loss exponent, taken to be 4 in these simulations. The energy consumed receiving a bit, e_{rx}, is simply E_{elec}.

The quote at the start of this chapter from Olariu and Stojmenovic related the assumption that in the corona model every node forwards its packets to a node in the next inward corona to the high density of sensor networks, therefore to test the assumption high densities were chosen. Kleinrock and Silvester showed that the optimal trade-off between connectivity and throughput is for every node to have six neighbours [KS78]. More neighbours reduces throughput but increases connectivity. This was later revised up to eight by Takagi and Kleinrock [TK84]. Stojmenovic and Lin found a similar result for sensor networks [SL01] citing eight to ten as high density.

To ensure that the networks examined were very high density, the number of neighbours was varied from a minimum of 25 neighbours per node up to 100 neighbours. Since the transmission range of each node is 10m the values used were 0.0785 nodes/m^2, 0.157 nodes/m^2, 0.2355 nodes/m^2 and 0.314 nodes/m^2 corresponding to 25, 50, 75 and 100 neighbours per node.

The analysis is validated by the results in Fig. 4.5 showing the number of nodes with the relay hole problem for a range of densities and network sizes. The results

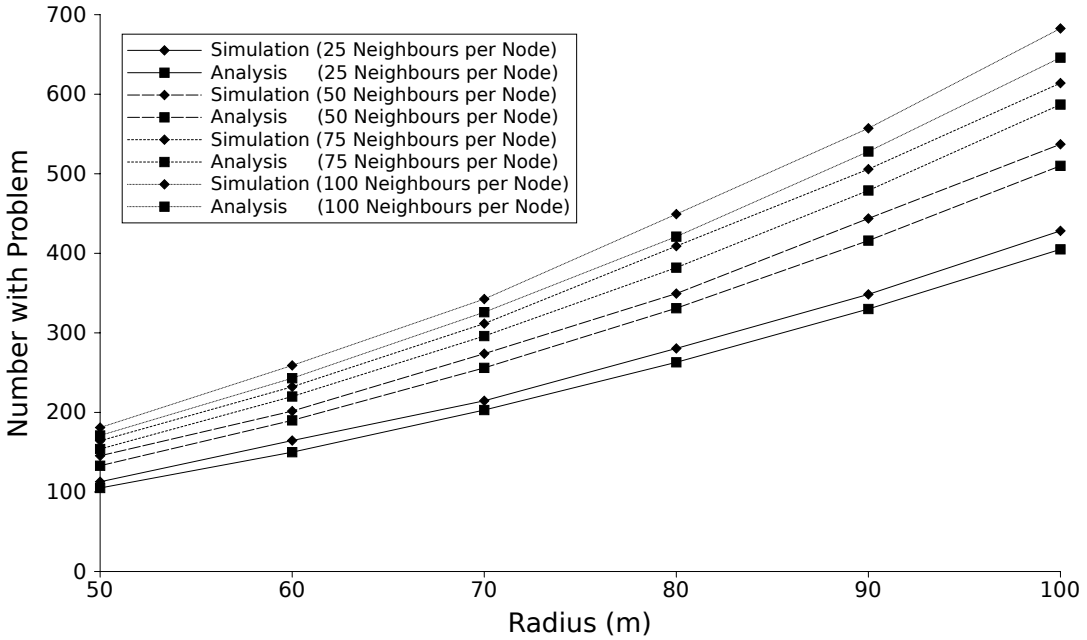


Figure 4.5: The difference between the analysis and simulation results are less than 10% which validates the analysis.

show that the difference between the simulation results and the analysis is less than 10% for all data points with the largest difference being 8.89% ($\pm 1.96\%$). In all cases the simulation and analysis are very strongly correlated ($r = 0.999$, $p \ll 0.001$).

Since in all cases the analytical results underestimate the simulation results, it seems likely that much of the difference is because the analysis does not include tertiary and other higher order knock-on effects of the problem.

The results also show that the number of nodes with the relay hole problem increases polynomially with radius ($r \geq 0.999$, $p \ll 0.001$ in all cases) but falls linearly with density ($r \geq 0.986$, $p \leq 0.014$).

The relationship between the number of nodes with the relay hole problem and both radius and density should not be looked at in absolute terms. This is because the total number of nodes in the network changes with radius and density. Rather, the proportion of nodes with the problem, as shown in Fig. 4.6 should be considered. They show that the proportion of nodes falls linearly but only slightly with radius ($r \leq -0.904$, $p \leq 0.013$) and falls polynomially with density ($r = -0.999$, $p < 0.001$).

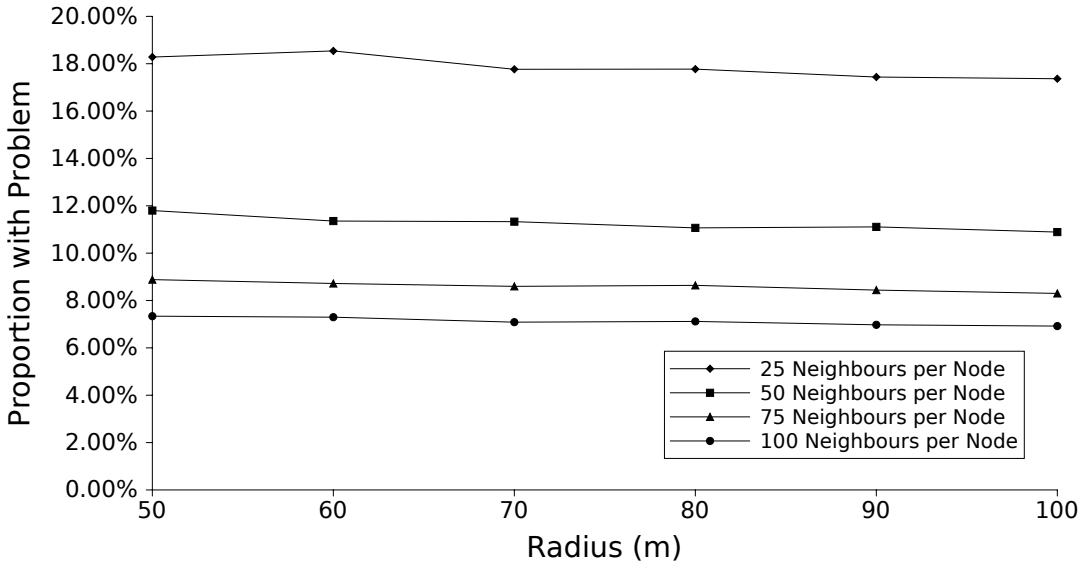


Figure 4.6: The proportion of nodes with the relay hole problem is almost invariant with radius but falls with density.

These results are consistent with the analysis which suggested, in equation (4.3), that the probability of a node having the problem is lower the further the node is away from the sink. Larger networks, with more coronas, have many outer coronas in which there is a lower chance of nodes having the problem and these coronas contain more nodes than the inner ones. This acts to lower the total proportion of nodes with the problem but the effect is small with the largest decrease being less than one percentage point.

The decrease with density is also predicted in equation (4.3) which shows that the probability of a node having the problem is polynomially related to the number of nodes in the network. This suggests that increasing density can reduce the problem. However, as Fig. 4.7 illustrates, increasing density brings diminishing returns. Even at the extremely high density of 100 neighbours per node on average 7% of nodes will have the relay hole problem.

4.3 Impact of the Relay Hole Problem

To show the impact of the relay hole problem, two deployment strategies are compared: uncontrolled deployment (UC) and fully-controlled deployment (FC). Uncontrolled deployment is when the nodes are positioned randomly in the network

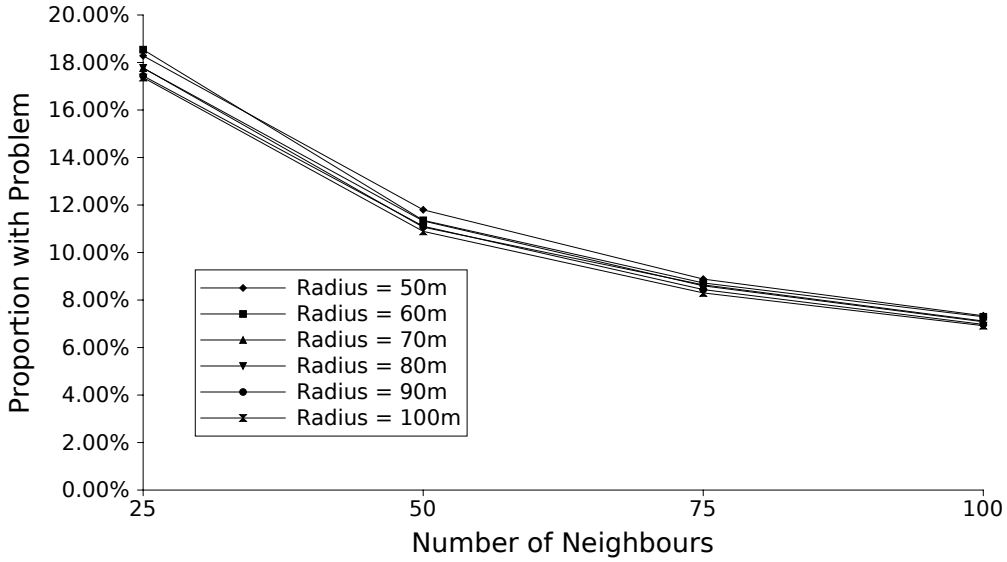


Figure 4.7: The benefit of increasing density suffers from diminishing returns.

area with the only condition being that the density is uniform. Fully-controlled deployment is when nodes are positioned in pre-determined locations to ensure that every node has at least one neighbour in its relay area. FC is the solution suggested in passing by Wu *et al.* which completely avoids the relay hole problem but, as discussed earlier, is expensive and sometimes infeasible [WCD08].

The impact of the problem is felt primarily through increased latency and reduced energy efficiency which can be measured by examining the network's residual energy when its lifetime ends (lifetime is measured as the time until the first node runs out of energy). Latency is measured as the average number of hops a packet in the network must travel through to reach the sink. Residual energy gives a view of the energy efficiency of the network. It measures the amount of energy left in the nodes when the first node dies, as a percentage of the initial network energy. In the type of networks considered in this thesis it has already been proved in Section 1.1 that the first node to deplete its batteries will be in the inner-most corona. The nodes in that corona cannot suffer from the relay hole problem because they are all, by definition, in direct communication with the sink and they collectively forward all packets from the network to the sink regardless of how many hops those packets traversed to reach the inner-most corona. Therefore, the lifetime of the network is independent of the relay hole problem and any changes to the value of residual energy between UC and FC

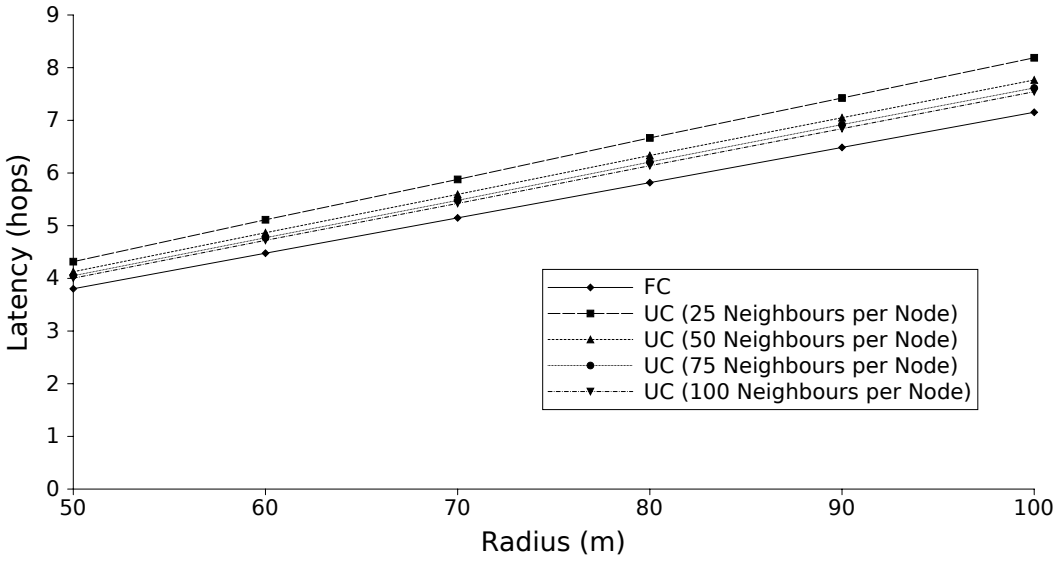


Figure 4.8: The relay hole problem causes an increase in latency which is worse at lower densities but still significant at high density.

cannot result from a change in lifetime and must represent only a change in energy efficiency during network operation.

Fig. 4.8 shows the results for latency. They show that the effect of the relay hole problem is to increase the average latency. The increase is independent of the radius of the network ($p \geq 0.044$) but is highly dependent on the density. The relationship is best described by an exponential correlation ($r = -0.999$, $p < 0.001$) which accords with the earlier result that increasing density reduces the proportion of nodes with the relay hole problem. However, as with the proportion so too with the latency and increasing density has diminishing returns.

The increase in latency is on average 14.24% ($\pm 0.245\%$) at the "low" density of 25 neighbours per node and falls to 5.42% ($\pm 0.065\%$) at 100 neighbours.

Fig. 4.9 shows the results for the residual energy. Although there is a statistically significant reduction in the amount of energy remaining when the first node depletes its batteries ($p \ll 0.001$), the loss of efficiency is not very large (the largest difference being 1.5 percentage points).

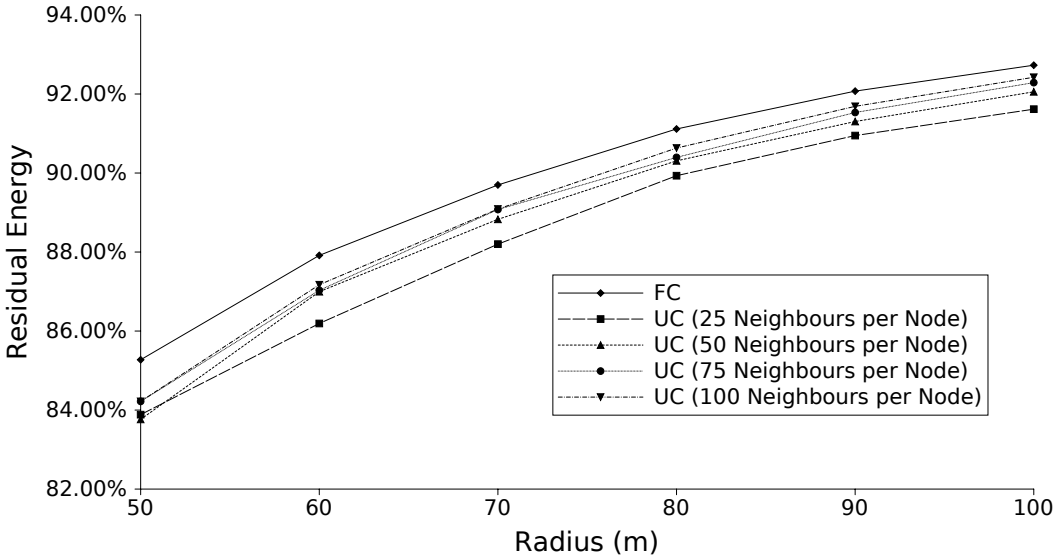


Figure 4.9: The relay hole problem also causes a statistically significant drop in energy efficiency which is revealed by a reduction in residual energy with no change to lifetime. However, the effect is very small.

4.4 Chapter Summary and Conclusions

In this chapter one of the underlying assumptions of the corona model, namely that nodes always forward their packets to a node in the next inward corona, has been examined. This assumption has been used to model the number of nodes in each corona by combining it with the assumption of uniform distribution to determine that the number of nodes per corona is proportional only to the area of the corona. This is then used as a fundamental element in the analysis of the energy hole problem by Li and Mohapatra [LM05, LM07] and others including Olariu and Stojmenovic [OS06].

However, the analysis in this chapter revealed that a problem, which was termed the relay hole problem, exists whereby the underlying assumption is not true. In fact, mathematical analysis showed that there is a significant probability that the portion of a node's reachable area that intersected the next inward corona may be empty of nodes. Moreover, even if the node was able to find a relay in the next inward corona, the relay itself may suffer from the relay hole problem which would have a knock-on effect to the node.

The analysis was confirmed through simulations and results showed that even

with a very high density of 100 neighbours per node (more than ten times the minimum required for guaranteed connectivity) 7% of the nodes suffer from the relay hole problem. Increasing the network density mitigates the problem to some extent but suffers from diminishing returns. Based on the mathematical analysis it is possible to show that at the incredibly high density of 1,000 neighbours per node there would still be 1.5% of nodes with the relay hole problem.

The relay hole problem not only causes an increase in latency but also makes it harder to predict the topology of the network because, despite a uniform distribution, not all nodes act as predicted. Whereas a node's physical position in the network area combined with the transmission range of all nodes suggests it ought to be a certain number of hops away from the sink, the relay hole problem means that it may be more hops away than predicted.

This is important for two reasons. Firstly, the novel routing protocols proposed in this thesis rely on the nodes acting as predicted in order to achieve high inner-corona balance. Although the effect of the relay hole problem on the balance produced by the proposed protocols was not quantified, the problem is likely to have had a statistically significant impact given the proportion of nodes that are affected by it.

More widely, the relay hole problem means that the analysis of the energy hole problem that has previously been carried out is not entirely accurate. Although the energy hole problem certainly exists, the analysis which assumes a simple relationship between the number of nodes in a corona and the corona's area needs to be revisited. Likewise, solutions to the energy hole problem that rely on this assumption also need to be revisited.

For example, the solutions that suggest a non-uniform distribution (see Section 2.2.5) all make this assumption in order to calculate the energy requirements of each corona and therefore the number of nodes needed in each to balance those requirements. However, the relay hole problem shows that simply placing a node in a physical corona does not guarantee that it will be topologically inside that corona. This calls into question the effectiveness of these solutions because in order to balance the workload they carefully calculate how many nodes must be in each corona and do so assuming that physical placement inside a corona is equivalent to topological placement. In light of the relay hole problem this approach to solving the energy hole problem needs revisiting.

The work presented in this chapter was published in [KF12a].

Chapter 5

Degree Balancing

5.1 Introduction

The main aim of this thesis is to propose fully distributed routing protocols for increasing inner-corona balance in static sensor networks. As discussed in Section 2.4, fully distributed protocols are those in which nodes select their own parent nodes based on information that originates with the neighbours they are in direct communication with. This limitation means that for distributed routing protocols, nodes cannot use the workload of their potential parents as a factor in their decision since information about workload requires gathering information from all descendants of a node. While it is certainly possible to gather the information, the cost of doing so is high as a large number of control packets must be passed through the network.

One measure that can be used is the degree of each node which is the number of neighbours that a node is directly connected to in the routing tree and is therefore local information. Since the protocols in this thesis produce a single, static routing tree, every node has only one parent and therefore a node's degree can be replaced with the number of children it has adopted.

Degree balancing is about minimising the variation in node degree among nodes in the same level of the routing tree. For the networks considered in this thesis, it is impossible for all nodes to have the same node degree because the number of nodes in neighbouring coronas varies through the network. Macedo analysed the

Level	Children per Parent	Number of Nodes in the Level
1	3	n
2	1.6666666667	$3n$
3	1.4	$5n$
4	1.2857142857	$7n$
5	1.2222222222	$9n$
6	1.1818181818	$11n$
7	1.1538461538	$13n$
8	1.1333333333	$15n$
9	1.1176470588	$17n$
10	1.1052631579	$19n$

Table 5.1: Values derived from equation (5.1) showing the average number of children per parent and the number of nodes in each level, where n is the number of nodes in level 1.

average number of children adopted per parent in each corona, c_i of a uniformly distributed network, C_i [Mac09] and his result is shown in equation (5.1). It should be noted that according to the corona model a node's corona number is equivalent to its level in a minimum-depth routing tree.

$$C_i = \frac{2i+1}{2i-1} \tag{5.1}$$

Macedo's analysis was designed to demonstrate that the node degree was not constant across the network; rather the average number of children per parent decreased further away from the centre. However it also shows that, except for the nodes in the first level, no node can have exactly the average node degree. This is because, as Table 5.1 illustrates, the average number of children per parent is a fraction for all levels except the first. The best that can be achieved is to minimise the variation in node degree such that some nodes have one child and some have two but none have three or zero children.

This form of load balancing is what I call *degree balancing* and involves minimising the variation in node degree among nodes of the same level. In this chapter, degree balancing is analysed with two aims. The first is to show that the probability of degree balancing alone resulting in perfect inner-corona balance is extremely low. In this context, "alone" means that the routing algorithm focuses only on the degree balance of a single level of the routing tree at a time and does not

consider inner-corona balance. This is the approach taken by the two proposed, fully distributed protocols, MBT [HCWC09] and MHS [CZYG10], which aim to maximise the degree balance at each level independently of the other levels and without consideration of inner-corona balance.

The second aim is to show that both of the above protocols show significant improvement over a simple distributed routing protocol that only constructs a shortest path routing tree without considering balance of any kind. This will establish the best performing of the two as a benchmark when considering the new protocols that will be proposed in Chapters 6 and 7.

5.2 Degree Balance and Inner-Corona Balance

As mentioned, Macedo's analysis shows that complete degree balance is impossible except in the first level of the routing tree. This is because the average degree is a fraction and nodes cannot adopt a part of a child. Degree balance is therefore about minimising the variation in node degree such that some nodes have one child and some have two children but none have three or zero. The only exceptions are the nodes in the first level which can have three children each. The problem with this is that it inevitably does leave some imbalance in the workload of nodes in the same level. Since each level is considered independently of every other and there is no information about workloads from other parts of the network, it is easy to see that this approach can also result in inner-corona imbalance.

The problem is illustrated by Fig. 5.1. The degree balance in the network shown is maximised and yet the inner-corona balance is not. It is trivial to see that if node B had adopted node C instead of node A, there would be both degree balance and perfect inner-corona balance. However, since each level's nodes are considering their adoptions independently of each other there is no reason for the assignment as shown to be avoided. In order to avoid the assignment, nodes A and B would have to know more about the other assignments in the network and that information is not usually local and is therefore not available in a distributed protocol.

For protocols that aim to maximise degree balance there is no way to guarantee

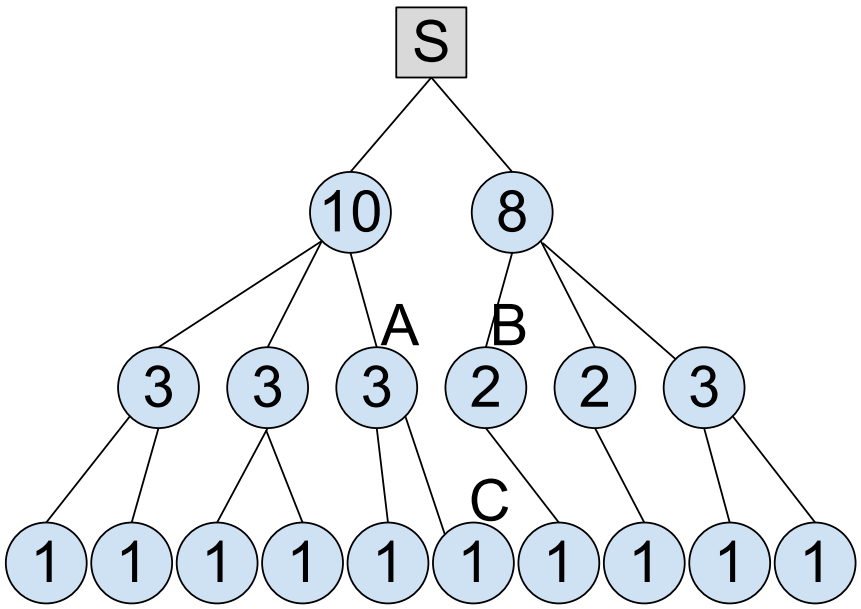


Figure 5.1: Since it is impossible for all nodes in most levels to adopt exactly the same number of children there will almost inevitably be some imbalance and therefore it is possible to have degree balance but not inner-corona balance.

complete inner-corona balance. Maximising degree balance might result in perfect inner-corona balance but, since inner-corona balance cannot be incorporated into the routing algorithm, it is only by chance that a protocol maximising degree balance would produce a routing tree with perfect inner-corona balance. It would therefore be useful to analyse the probability that this would happen. If it turns out that the probability is high then degree balancing is a promising approach. However, if, as seems likely, the probability is extremely low then this approach is far less promising.

Perfect inner-corona balance results when no top subtree (that is, a subtree rooted at a level one node) contains more nodes than any other top subtree. In order for a degree balanced tree to also have perfect inner-corona balance the nodes in each level that have adopted two children each must be drawn evenly from the top subtrees. In order to facilitate the analysis of the probability that this happens some new terminology needs to be defined.

In a degree balanced tree nodes either adopt one, two or three children. Let the terms *singles*, *doubles* and *triples* refer to these nodes respectively. With reference to Table 5.1 it is obvious that every level one node ought to be a triple because

the average number of children per parent for level one is three. For the other levels it is not immediately obvious how many singles and doubles there should be from the children per parent ratio. However, examining the ratio between the number of nodes in a given level and the number in level one, R_i, gives a simple solution.

The ratio, given below in equation (5.2), is easily derived from Macedo's equation above (5.1) and is shown in Table 5.1 in the third column as the coefficient of n in the number of nodes in each level. It is clear that in each level of the routing tree there are $2n$ more nodes than in the previous level where n is the number of nodes in the first level. Therefore, in each level (except the first) there should be $2n$ doubles and all the other nodes should be singles. This is because if all nodes in a level were singles then all but $2n$ nodes in the next level would be adopted. Having $2n$ nodes as doubles means that there are an extra $2n$ adoption "slots" available and so all nodes in the next level are able to be adopted.

$$R_i = 2i - 1 \tag{5.2}$$

As the number of doubles in each level can be related to the number of nodes in the first level it is simple to see how perfect degree balance could lead to perfect inner-corona balance. Perfect inner-corona balance results from all top subtrees having the same number of descendants and since each node in level one is the root of one of the top subtrees there are obviously n such top subtrees. To get perfect inner-corona balance there must be no deviation between the number of descendants of these top subtrees which means that each must have the same number of doubles as each other in every level since only the doubles cause variation. There would be no variation at all if all the nodes were singles but this would prevent many nodes connecting to the routing tree and result in very low connectivity. Thus, to ensure perfect inner-corona balance from degree balance, the $2n$ doubles in every level must be drawn evenly from the n top subtrees, meaning that exactly two doubles must come from each top subtree. This fact makes it possible to analyse the probability of this happening by chance - it is the probability that in every selection, precisely two doubles come from each top subtree.

Equation (5.3) gives the probability that, in level l_i of the routing tree, the doubles are assigned perfectly to the top subtrees such that level l_i contributes to perfect

inner-corona balance, β_i. The derivation of this probability can be found in Appendix A.

$$P(\beta_i) = \frac{(2i-1)^n(2i-2)^n(2in-3n)!}{(2in-n)!}\frac{(2n)!}{(2!)^n} \tag{5.3}$$

The probability that the entire tree will be balanced is the product of balance at every level. However, the first level is excluded because balance is inherent and the last level is also excluded because the status of a node in that level as a single or double does not affect the balance of the tree as no nodes in the outer-most level adopt children. The probability of a perfectly balanced tree occurring from random selection of doubles is:

$$P(\beta) = \prod_{i=2}^{k-1} P(\beta_i) \tag{5.4}$$

This probability quickly becomes very small. For example, when there are ten subtrees and five levels, the probability of the routing tree being perfectly balanced is $4.2967 \times 10^{-9}\%$.

5.2.1 Simulation Validation

To validate the above analysis, a theoretically perfect, degree balancing protocol was simulated, one that ensured that in every level the variation in node degree was at most one. The centralised algorithm is termed Centralised Degree Balance (CDB) and guarantees that perfect degree balance is obtained. The algorithm is designed to work in a scenario with two unrealistic assumptions. The first is that the nodes are deployed in perfect accordance with Macedo's analysis as specified in equation (5.1). The second is that each node can communicate directly with every node in its neighbouring coronas. This ensures that there are no routing holes or relay holes (see Chapter 4) and that there is complete freedom to choose which nodes are children of which parents.

CDB is designed only to ensure perfect degree balance and so does not consider inner-corona balancing at all. The algorithm divides the nodes into their levels based on which corona they are in. In the first level all nodes are assigned the

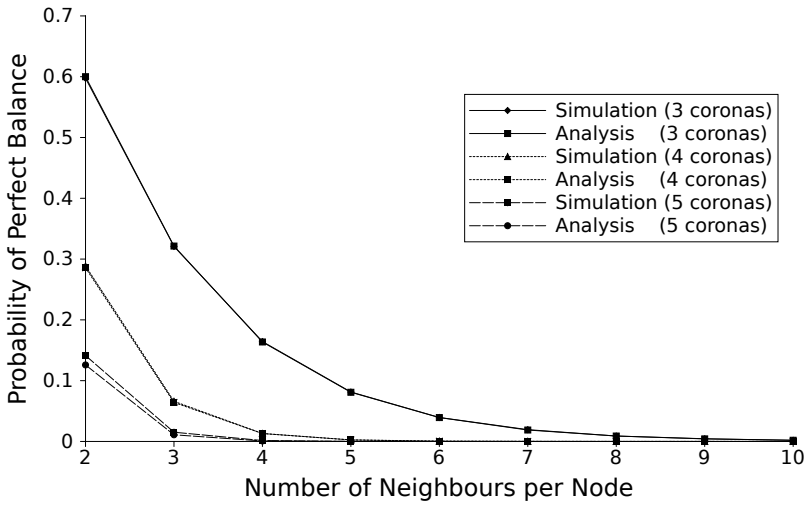


Figure 5.2: The simulation results verified the analysis showing that the probability of producing a perfectly balanced routing tree with randomly assigned doubles quickly approaches zero.

role of triples, that is they are each allowed to adopt up to three children. The algorithm thereafter works level by level assigning nodes in the next level as children of the nodes in the level being processed. The assignment respects the roles of the nodes as singles, doubles or triples such that no node is assigned more children than its maximum allows. Once a set of nodes have been assigned as children, $2n$ of them are randomly selected to be doubles and all the rest are singles (where n is the number of nodes in the first level). This is in line with the analysis above that showed that for perfect degree balance every level (except the first) should consist of $2n$ doubles with all the rest being singles. Algorithm 5.1 gives the pseudocode for the CDB algorithm. The CDB algorithm is also used to consider the performance of degree balancing as an approach in Section 5.3.2.

In order to find the probabilities of perfect balance, the values of n and the number of coronas in the network were varied and the proportion of runs that had perfect balance were calculated. Because the probabilities are sometimes very low, the simulations involved 100,000 runs for each configuration. The results are shown in Fig. 5.2.

The experimental results verify that the analysis is correct. For a small network of only three coronas the discrepancy between the simulations and analysis was on average 1% ($\pm 1.24\%$) and the two are almost perfectly correlated ($r > 0.999$,

Algorithm 5.1 CentralisedDegreeBalance

```
1: function CentralisedDegreeBalance(nodes,sink)
2:     for all node ∈ nodes do                          ▷ Initialise the nodes
3:         node.parent = NULL
4:         node.sinkDist = √((node.X − sink.X)² + (node.Y − sink.Y)²)
5:         node.level = 1 + ⌊node.sinkDist/transmissionRange⌋
6:         node.maxChildren = 1
7:         levels[node.level].put(node)
8:         if node.level == 1 then
9:             node.maxChildren = 3
10:            node.setParent = sink
11:        end if
12:    end for
13:    for all level ∈ levels do
14:        for i ← 1, 3 do                              ▷ Assign children to parents
15:            for all parent ∈ level do
16:                if parent.numberChildren < parent.maxChildren then
17:                    for all child ∈ levels[level+1] do
18:                        if child.parent==NULL then
19:                            child.parent = parent
20:                            parent.numberChildren++
21:                        end if
22:                    end for
23:                end if
24:            end for
25:        end for
26:        numDoubles = 2×levels[1].size
27:        for i ← 1, numDoubles do                     ▷ Assign doubles in next level
28:            done = false
29:            while !done do
30:                futureParentIndex = rand(0,levels[level+1])
31:                futureParent = levels[level+1].get(futureParentIndex)
32:                if futureParent.maxChildren == 1 then
33:                    futureParent.maxChildren = 2
34:                    done = true
35:                end if
36:            end while
37:        end for
38:    end for
39: end function
```

$p \ll 0.001$). As the number of coronas increases the probabilities fall and the discrepancies increase. Nevertheless, the correlation between the analysis and simulation results remain extremely high ($r > 0.999$, $p \ll 0.001$).

It is clear, then, that the probability of achieving perfect inner-corona balance through degree balancing is extremely small. This suggests that it may not be an effective method for maximising inner-corona balance and this will be examined in the next section.

5.3 Degree Balancing as an Approach

Since the degree balancing approach cannot guarantee perfect inner-corona balance even in idealistic scenarios, it is unlikely to result in the highest possible inner-corona balance in more realistic circumstances. Nevertheless, the approach should be able to produce higher balance than other routing algorithms that do not make any attempt at any kind of load balancing. Therefore, the best performing of the two existing distributed routing algorithms that aim to maximise degree balance, namely MBT and MHS (see Section 2.4, pages 55-58), should serve as a useful benchmark for the new protocols that will be proposed in the next two chapters. Their usefulness as a benchmark, however, depends on their ability to significantly outperform a simple routing algorithm.

In this section the performance of degree balancing as an approach is initially considered in ideal circumstances as proof-of-concept that MBT and MHS might outperform simpler algorithms. Later the performance of those algorithms are directly compared to a simple, distributed algorithm that does not consider load balancing.

5.3.1 Baseline Algorithms

There are two potential routing algorithms that can serve as a baseline. The first is first-heard-from (FHF) described first by Beaver *et al.* [BSLC04]. In this algorithm the nodes initialise their level of the tree to ∞ while the sink sets its own level to zero. The sink then broadcasts a beacon containing its level and this starts the process of building the routing tree. Any node that receives a beacon

compares the contained level to its own and, if the level is more than one level lower (i.e. the node would decrease its level by becoming a child of the sender) then it changes its level and selects the sender of the beacon as its parent. Once a node changes its level it broadcasts its own beacon.

The FHF algorithm is simple and usually requires only one packet to be transmitted per node. However, there may be situations in which a node first hears a beacon from a node that is more hops away from the sink than other potential parents and then later hears a beacon from a parent with fewer hops to the sink. In this case the node will switch levels twice and broadcast a second beacon (which may result in other nodes transmitting extra beacons as well). In the networks considered in this thesis this is unlikely as there is no interference and no collisions.

An alternative is based on the Greedy Forwarding method described by Karp and Kung [KK00]. In greedy forwarding, nodes are aware of their positions and those of their neighbours and each packet in the network contains the position of the final destination. Every intermediate node selects its neighbour that is closest to the final destination as the next hop.

In a data-gathering network, with static nodes, where all packets are destined for the sink, the next hop in greedy forwarding will always be the same node. Therefore for these networks greedy forwarding can be turned into a routing tree which is called simply shortest-path tree (SPT) in this thesis. Similar to the FHF tree, the nodes initialise their levels to ∞ and await beacons. However, the beacons contain not only the sender's level but also its position, obtained through GPS or some other method. When a node receives a beacon it stores the information it contains, waiting for some predefined time while it collects beacons from all its neighbours. Then it chooses the node that is closest to the sink as its parent.

The SPT is likely to be more balanced than FHF because under FHF the first node to transmit its beacon will adopt all its neighbours simply by virtue of being first. However, under SPT the choice of parent is based on distance and, since the nodes are uniformly distributed, every node will have a set of potential parents at different distances. It is unlikely that one node will be the neighbour closest to the sink of a large number of nodes. Rather, it seems more likely that under SPT a node will be the preferred parent of only a small number of others which would

lead naturally to some form of load balancing. Therefore, the SPT approach is chosen as the baseline in this chapter.

5.3.2 Degree Balancing in an Ideal Scenario

Before considering the performance of the proposed protocols, MBT and MHS, the degree balancing approach is considered in an ideal (and unrealistic) scenario by comparing the CDB algorithm, described above in Section 5.2.1, to the baseline algorithm. The ideal scenarios is as described earlier, also in Section 5.2.1, that the nodes are distributed in perfect accordance with Macedo's analysis in equation (5.1) and that every node is in direct communication with every other node in its neighbouring coronas. These knowingly unrealistic assumptions isolate the performance of degree balancing from any complications arising from uneven deployment and routing or relay holes.

However, the SPT algorithm as described above does not work with the assumption that nodes can communicate directly with every node in their neighbouring coronas. Therefore, for comparison a centralised random (CR) algorithm was used in which children select their parents randomly from the nodes in the previous corona without assigning any limits to the number of children nodes may adopt. The pseudocode for CR is shown in Algorithm 5.2.

The results in the remainder of this chapter are from simulations under various configurations. The radius of the network was varied from 50m to 100m and the density was also varied from ten neighbours per node to twenty. Each configuration was repeated 25 times and the results averaged.

The results show that for every configuration, the CDB algorithm generated a routing tree with perfect degree balance. That is, the difference between the number of children adopted by the node that adopted the most and the node that adopted the fewest was exactly one for all levels except the first level where all nodes had exactly the same number of children.

Fig. 5.3 shows the results for inner-corona balance which show that even though CDB never achieves perfect inner-corona balance, it nevertheless produces extremely high inner-corona balance. The balance falls slowly with radius ($r = -0.991$, $p = 0.00012$) decreasing from an average of 0.971 (± 0.002) with a 50m

Algorithm 5.2 CentralisedRandom

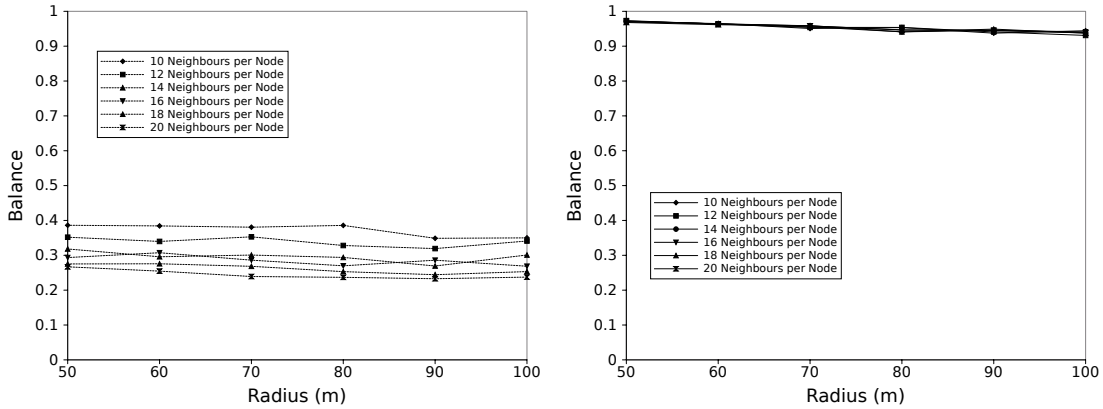
```
1: function CentralisedRandom(nodes,sink)
2:     for all node ∈ nodes do                          ▷ Initialise the nodes
3:         node.parent = NULL
4:         node.sinkDist = √((node.X − sink.X)² + (node.Y − sink.Y)²)
5:         node.level = 1 + ⌊node.sinkDist/transmissionRange⌋
6:         levels[node.level].put(node)
7:         if node.level == 1 then
8:             node.setParent = sink
9:         end if
10:    end for
11:    for all level ∈ levels do                        ▷ Assign children to parents
12:        for all child ∈ levels[level+1] do
13:            parentIndex = rand(0,level.size)
14:            parent = level[parentIndex]
15:            child.parent = parent
16:        end for
17:    end for
18: end function
```

radius to 0.938 (± 0.005) with a radius of 100m. However, it is not significantly dependent on the density ($p = 0.945$).

This makes sense because as the number of coronas increases so does the impact of an unbalanced assignment. Remember that ideally only two doubles would be chosen from each subtree in each level. If an extra double is chosen from one subtree then it will result in that subtree having more work because the double adopts an extra child. The problem is exacerbated by having more levels because that extra child goes on to adopt more children and so the difference between the overloaded subtree and the others grows larger.

The lack of correlation with density is initially a little surprising. One might expect that as the density increases the inner-corona balance would fall somewhat because the probability of doubles being drawn precisely two from each subtree would fall. Looking at equation (5.3), the probability of a perfect assignment at each level depends on the number of subtrees which is directly related to the density.

It is apparent, however, that the probability of perfect assignment is negligible in any case and therefore increasing density has little practical effect. For example,



(a) CR (b) CDB

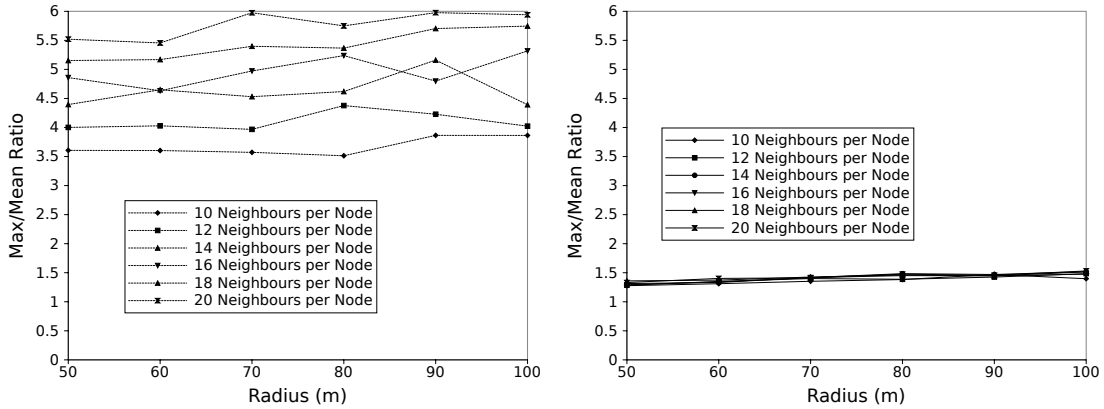
Figure 5.3: Simulation results show that the balance achieved by a centralised balancing algorithm in ideal circumstances falls slightly with radius but does not vary with density. In all cases though it significantly outperforms a random assignment.

when $i = 4$ and $n = 10$ equation (5.3) evaluates to 1.98×10^{-83}. Increasing n to 20 reduces that probability to 5.78×10^{-198}. This is clearly a big change but since the probability was so low to begin with it is evidently not resulting in any noticeable effect.

CDB improves the inner-corona balance by between 147.11% (±21.79%) and 305.60% (±28.10%) compared to CR. The improvement increases with radius ($r = 0.842$, $p = 0.036$) and with density ($r = 0.995$, $p \ll 0.001$). The increase in improvement with density results from the decrease in performance of CR with density ($r = -0.982$, $p = 0.0005$) because the performance of CDB is invariant with density.

Fig. 5.4 shows the results for the max/mean ratio which indicates lifetime. The results show that the lifetime under CDB would be much higher than under CR. Using CDB, the max/mean ratio varies from a high of 1.53 (±0.065) to a low of 1.28 (±0.032). The ratio increases with the radius ($r = 0.985$, $p = 0.00034$) and density ($r = 0.898$, $p = 0.015$).

Given that the balance metric does not vary with density this last result appears surprising. The most likely explanation is that there is some small decrease in balance as a result of increasing density but this decrease is hidden because the balance metric considers the sizes of all the subtrees. The max/mean ratio,



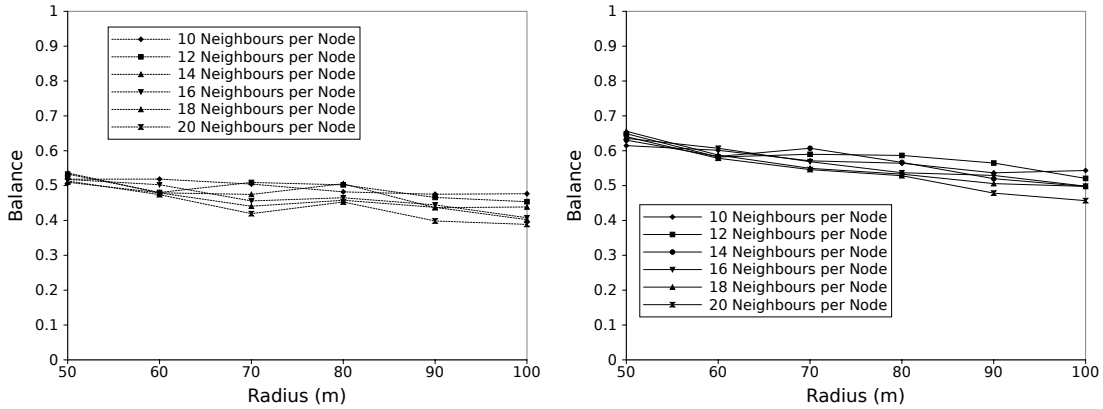
(a) CR (b) CDB

Figure 5.4: Simulation results show that the max/mean ratio achieved by a centralised balancing algorithm in ideal circumstances varies little with density but increases with radius. In all cases it is significantly lower than when using the centralised random algorithm.

however, only considers the size of the largest subtree which makes it more sensitive to small changes. In fact, when the radius is constant the mean is entirely invariant with density because doubling the density doubles both the number of nodes in the network and also the number of subtrees. Therefore, in effect, max/mean is only measuring the size of the largest subtree and will reveal even small changes.

Additional simulation experiments with a radius of 50m and 60m appear to confirm this explanation. With a radius of 50m, the size of the largest subtree shows a strong correlation with density ($r = 0.873$, $p = 0.023$) and this correlation is even more pronounced with a radius of 60m ($r = 0.895$, $p = 0.016$). However, the growth is very small: with a radius of 50m the range of values is 31.92 (± 0.799) to 33.28 (± 0.759) and with a radius of 60m is 47.2 (± 1.94) to 50.48 (± 1.67). Such small changes would be evident in the max/mean ratio which is sensitive only to these figures but would be lost in the balance metric which is calculated based on the sizes of all subtrees.

The max/mean ratio under CDB is between 60.65% ($\pm 3.98\%$) and 76.37% ($\pm 1.66\%$) lower than under CR. The improvement shows a statistically significant correlation with density ($r = 0.990$, $p = 0.0016$) but not with radius ($r = -0.81$, $p = 0.05$).



(a) SPT (b) MBT

Figure 5.5: The MBT algorithm produces between 13% and 30% more balance than SPT and the effect increases with density. The improvement appears to fall with increasing radius but that result might be due to simulation error.

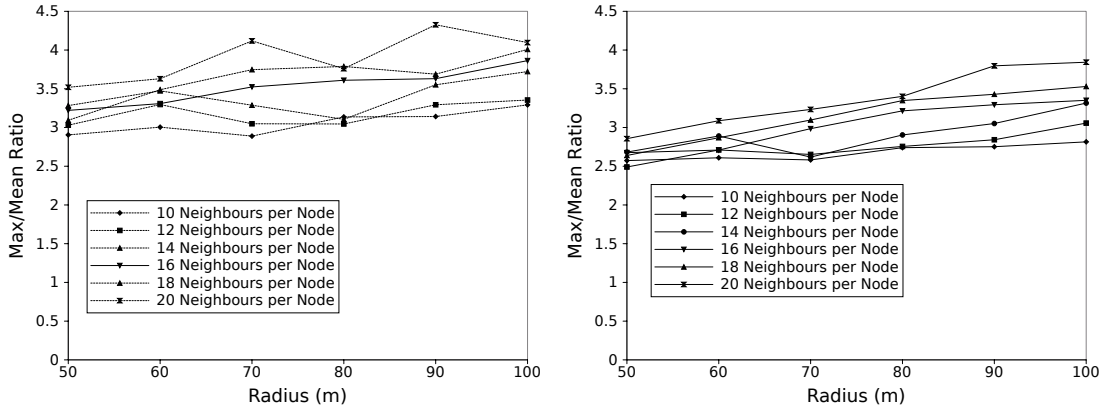
From these initial results it is clear that the degree balancing approach can generate higher inner-corona balance than a random assignment which suggests that the distributed degree-balancing algorithms of MBT and MHS would offer a useful benchmark for the new protocols to be proposed in Chapters 6 and 7.

5.4 Performance of MBT and MHS

The performance of MBT and MHS are compared first to SPT and then to each other. Fig. 5.5 shows the balance achieved by MBT compared to SPT. MBT shows an improvement of between 12.90% (±7.03%) and 30.13% (±14.27%) over SPT. The improvement falls with radius but the effect is doubtful ($r = -0.73$, $p = 0.096$), however it increases with density ($r = 0.926$, $p = 0.0079$).

Fig. 5.6 shows the results for the max/mean ratio and a general trend can be seen showing that MBT generally has a lower max/mean ratio than SPT and that the ratio increases with radius. In fact, the MBT algorithm shows a reduction of between 12.66% (±5.05%) and 21.49% (±5.71%) over SPT. The improvement falls with radius ($r = -0.84$, $p = 0.036$) but shows no statistically significant correlation with density ($r = 0.494$, $p = 0.319$).

Fig. 5.7 shows the balance achieved by MHS compared to SPT. MHS shows an



(a) SPT (b) MBT

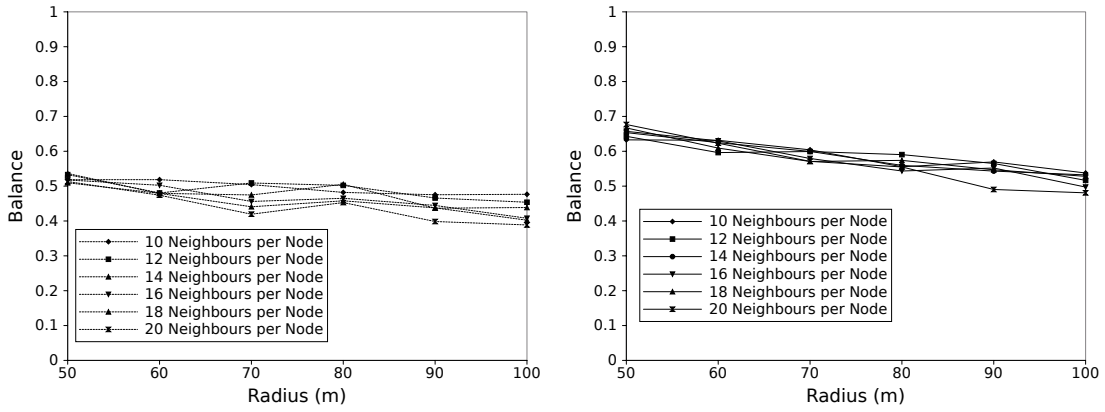
Figure 5.6: The max/mean ratio is lower under MBT than SPT by between 13% and 22%. The improvement falls with radius but shows no statistically significant relationship with density.

improvement of between 10.74% (±8.45%) and 36.13% (±15.39%) over SPT. The improvement shows no statistically significant relationship with radius ($r = 0.463$, $p = 0.137$), however it increases with density ($r = 0.975$, $p = 0.0009$).

Fig. 5.8 shows the results for the max/mean ratio. A general trend can be seen showing that MHS generally has a lower max/mean ratio than SPT and that the ratio increases with radius. In fact, the MHS algorithm shows a reduction of between 3.16% (±4.93%) and 24.32% (±4.78%) over SPT. The improvement generally falls with radius but the significance of this result is doubtful ($r = -0.797$, $p = 0.057$). The results shows no statistically significant correlation with density ($r = 0.718$, $p = 0.108$).

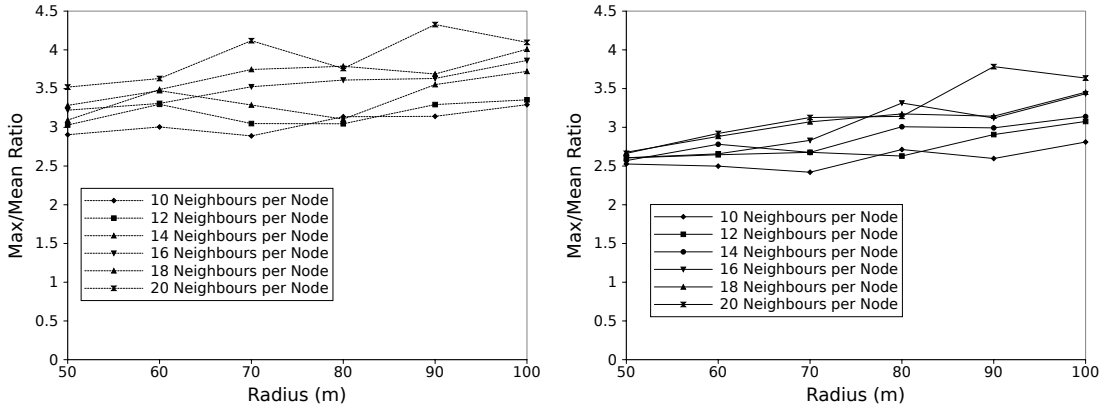
These results illustrate that both MBT and MHS are improvements over SPT and both are therefore candidates for a useful benchmark. Comparing the two directly shows that they are both similar in terms of balance but MHS usually outperforms MBT by a small amount, up to 8.13% (±3.4%). There is no statistically significant correlation between the improvement of MHS over MBT with radius ($p = 0.794$) and it is doubtful that there is a correlation with density ($p = 0.095$). If such a correlation were to exist, however, then it would demonstrate that the improvement increases with density ($r = 0.736$).

A similar set of results are found for the max/mean ratio with MHS usually having a slightly lower ratio than MBT by up to 8.33% (±6.16%) although MBT



(a) SPT (b) MHS

Figure 5.7: The MHS algorithm produces between 11% and 36% more balance than SPT and the effect increases with density but is independent of radius.



(a) SPT (b) MHS

Figure 5.8: MHS reduces the max/mean ratio by between 3% and 24% compared to SPT but this improvement appears independent of both density and radius.

shows an improvement over MHS of 4.59% (±7.27%) in one case. Overall the difference between them shows no correlation with radius ($p = 0.81$) or with density ($p = 0.304$).

Since the MHS algorithm shows better performance than MBT more often than not and has far lower overhead, it can serve as a useful benchmark when considering the proposed routing algorithm in the next two chapters.

5.5 Conclusion

The ultimate aim of routing protocols designed to maximise load balancing is to maximise network lifetime by mitigating the energy hole problem. In Section 1.1 it was shown that in terms of network lifetime there was no advantage to intra-corona balance over inner-corona balance. In this chapter it has been shown that degree balance cannot guarantee to produce the same lifetime as intra-corona or inner-corona balance, even in the most perfect of circumstances.

Of the existing distributed protocols for load balancing in sensor networks the best performing were ones that maximised degree balancing, such as MBT [HCWC09] and MHS [CZYG10] and it is therefore perhaps surprising that their optimal behaviour cannot guarantee to provide the same lifetime as the optimal behaviour of approaches that maximise intra-corona or inner-corona balance. What this reveals is that some amount of global knowledge is needed whether this knowledge is gathered during tree construction or incorporated into the protocol *a priori*.

The result also suggests that tackling the problem of lifetime maximisation through degree balancing is capable of only producing limited benefits and suggests that future research efforts should look elsewhere for solutions. For this reason new protocols that aim to maximise inner-corona balance as a means of lifetime maximisation are proposed in the next two chapters and these protocols form the major contribution of this thesis.

It is important to note, however, that despite the inability of degree balancing to ensure perfect balance, the results in this chapter nevertheless show that both of the existing degree balancing algorithms (MBT and MHS) are significant improvements over a simple shortest path routing tree. As such either could serve

as a useful benchmark for the new protocols. However, since MHS more often than not outperforms MBT and has far lower overhead, I have chosen MHS as the benchmark.

Chapter 6

Role Based Routing

The preceding chapter showed that the degree balancing approach could not guarantee perfect inner-corona balance even in the most ideal of circumstances and it is therefore unlikely to be as effective in more realistic scenarios as an approach that can make that guarantee. This chapter proposes the first of two novel routing algorithms that are designed specifically to maximise inner-corona balance. To the best of my knowledge these are the first such algorithms to have been proposed.

The next section will describe the theory underpinning role based routing. In Section 6.2 the theory will be validated with a centralised algorithm in ideal circumstances proving that role based routing can guarantee perfect inner-corona balance. However, the sensitivity of this approach to the ideal circumstances will be revealed. Nevertheless, a distributed implementation of the approach will be shown to outperform the benchmark (MHS) in Section 6.3.

6.1 Theory of Role Based Routing

Recall that Macedo showed that in a uniformly distributed network, the average number of children per parent in a given corona C_i is [Mac09]:

$$C_i = \frac{2i+1}{2i-1} \tag{6.1}$$

This is based on the observation that the number of nodes in corona i is $(2i-1)n$ where n is the number of nodes in corona one. Thus the difference in the number of nodes in neighbouring coronas is always $2n$:

$$\begin{aligned} diff &= (2i+1)n - (2i-1)n \\ &= (2in+n) - (2in-n) \\ &= 2n \end{aligned} \tag{6.2}$$

There are two important implications from this result. The first is that it is impossible to assign all the nodes in one corona as children of the parents in the inner corona in a way that results in all the parents having the same number of children. The most that can be achieved is that the largest difference in the number of children adopted by parents is one.

The second implication is that when minimising the difference in the number of children adopted, all parent nodes will have one child each but $2n$ nodes will need to adopt a second child. The fact that the number of nodes needing to adopt a second child is a constant and proportional to the number of nodes in the innermost corona leads to a possible method for providing high global balance in a distributed fashion.

Each of the nodes in the innermost corona become the root of a subtree and high inner-corona balance is achieved when these subtrees are the same size. From equation (6.1), each of the nodes in the innermost corona will adopt three children each. Thereafter, all nodes will adopt one child and $2n$ nodes in each corona will adopt a second one. A method for maintaining high inner-corona balance is to make sure that the $2n$ nodes that adopt extra children are evenly apportioned among the n subtrees, preferably so that there are exactly two such nodes from each subtree.

I propose a form of routing called ROle BAsed Routing (ROBAR) in which nodes are assigned a specific role relating to the number of children they may adopt. There are three roles: *triples*, *doubles* and *singles*. As the names imply, the triples may adopt up to three children each, the doubles up to two and the singles may only adopt one child. Thus the nodes in the innermost coronas are the only triples

in the network. In every other level there are $2n$ doubles and all the other nodes are singles.

ROBAR aims to ensure that the $2n$ doubles are evenly spread among the subtrees by controlling which nodes may be doubles. A routing tree which successfully implemented the role based routing approach would have the following four properties:

1. All triples would have three children of which two are doubles
2. All doubles would have two children of which only one is a double
3. All doubles are children of either a double or triple
4. All nodes that are not doubles or triples are singles

The first property applies to nodes in the first level of the tree which are the only triples in the network. There are n nodes in the first level but in the second level there must be $2n$ doubles. If every triple selected precisely two of its children to be doubles then there will be the requisite number of doubles and they will have been evenly selected from the n subtrees. Thereafter, the number of doubles must remain constant and this is achieved when every double select only one of its children to be a new double (property 2) and doubles are not selected by singles (property 3). These two properties combine to ensure that if there are $2n$ doubles in one level there will be $2n$ doubles in the next. Moreover, since every double has one child that is a new double and the original doubles were properly shared among the subtrees, the new doubles will also be properly shared among the subtrees. The final property states simply that every node has a role.

These properties can be brought about in a straightforward manner in a distributed algorithm. The triples in the innermost corona appoint two of their children to be doubles. Once those doubles have adopted their two children they appoint only one child to be a double. In this way there are always $2n$ doubles and it is always the case that exactly two are drawn from each subtree.

In the next section, simulations are used to prove that role based routing can guarantee perfect inner-corona balance in ideal circumstances which makes it likely that the approach can produce higher balance in more realistic scenarios than the benchmark which cannot make this guarantee.

6.2 Theory Validation

In this section a centralised implementation of the role based routing approach is used to prove that in ideal circumstances the approach can guarantee perfect inner-corona balance. The ideal circumstances are the same as those described in the previous chapter, namely that the distribution of the nodes follows the theory as described in equation (6.1) perfectly and that (unrealistically) all nodes can communicate directly with every node in its neighbouring coronas.

The centralised implementation is called CROBAR (Centralised ROle BAsed Routing) and is specified in algorithm 6.1. During the first part of the algorithm the nodes are initialised and assigned to levels. The nodes in the inner-most corona are set as children of the sink and assigned the role of triples by setting their maximum number of children to three.

In the next phase the tree is built up level by level with nodes in one level adopting children in the next. Rather than all doubles or triples adopting two or three children in one go, all nodes adopt one child at a time before doubles and triples receive a chance to adopt a second child and finally triples receive a chance to adopt a third child. Since all nodes in one corona are assumed to be in direct communication with all nodes in the next corona the choice of which child a parent should adopt is immaterial and so children are adopted sequentially. Finally, in the final phase once all nodes in one level have been adopted some are appointed as doubles.

Simulations were run using the same configurations described in the previous chapter. For every configuration CROBAR produced perfect inner-corona balance and a max/mean ratio of exactly one. These results confirm the theory that role based routing can guarantee perfect balance in idealised circumstances.

Role based routing, however, is very sensitive to these ideal circumstances. The theory behind it relies on there always being precisely $2n$ doubles in each level of the routing tree which is only true if the distribution of nodes follows equation (6.1) precisely. If the distribution differs then not only can role based routing not guarantee perfect balance but not all the nodes in the network will be able to connect to the tree. Some levels may have fewer nodes than predicted which will mean that some doubles are unable to adopt two children leading to imbalance. On the other hand, some level may have more nodes than predicted and those

Algorithm 6.1 CROBAR

```
1: function CROBAR(nodes,sink)
2:     for all node ∈ nodes do                          ▷ Initialise the nodes
3:         node.parent = NULL
4:         node.sinkDist = √((node.X − sink.X)² + (node.Y − sink.Y)²)
5:         node.level = 1 + ⌊node.sinkDist/transmissionRange⌋
6:         node.maxChildren = 1
7:         levels[node.level].put(node)
8:         if node.level == 1 then
9:             node.maxChildren = 3
10:            node.setParent = sink
11:            node.setDouble( true )
12:        end if
13:    end for
14:    for all level ∈ levels do
15:        for i ← 1, 3 do                              ▷ Assign children to parents
16:            for all parent ∈ level do
17:                if parent.numberChildren < parent.maxChildren then
18:                    for all child ∈ levels[level+1] do
19:                        if child.parent==NULL then
20:                            child.parent = parent
21:                            parent.numberChildren++
22:                        end if
23:                    end for
24:                end if
25:            end for
26:        end for
27:        for all parent ∈ level do                    ▷ Assign doubles
28:            if parent.isDouble() then
29:                parent.child(0).setDouble(true)
30:                parent.child(0).maxChildren = 2
31:                if parent.level == 1 then
32:                    parent.child(1).setDouble(true)
33:                    parent.child(1).maxChildren = 2
34:                end if
35:            end if
36:        end for
37:    end for
38: end function
```

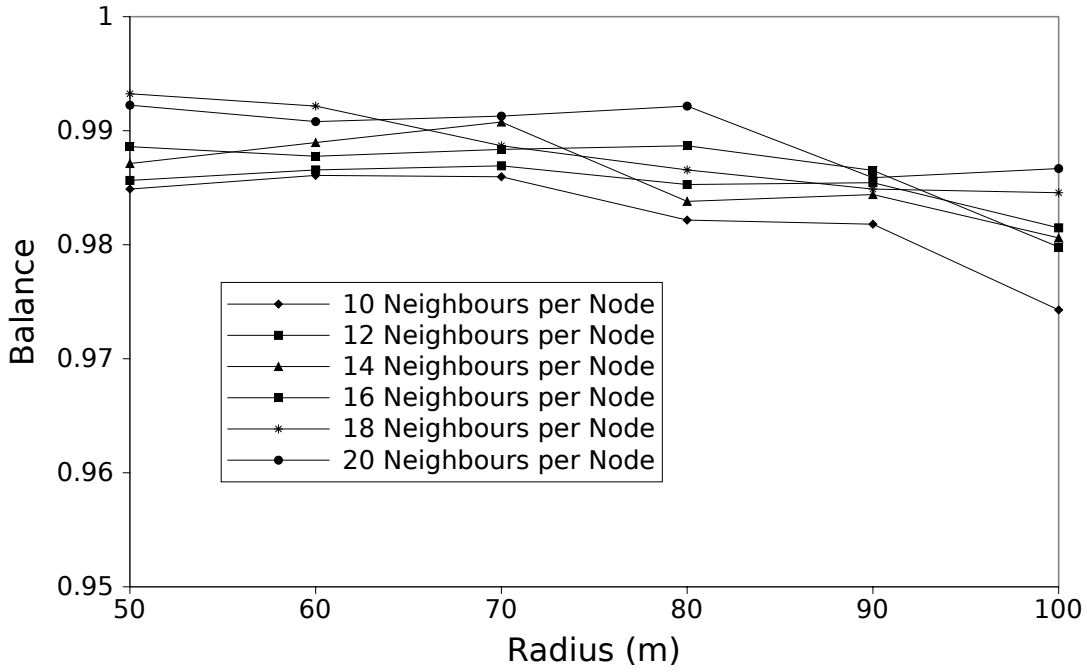


Figure 6.1: When a uniform random distribution of nodes is used instead of one matching equation (6.1) the balance becomes less than one.

extra nodes will not be adopted because the nodes in the next inward corona will have filled their quotas.

To show this, simulations were run as before but nodes were distributed uniformly. The results show that balance and connectivity fall when moving from the ideal distribution that follows equation (6.1) to a random uniform distribution which may not match that equation. Fig. 6.1 shows the results for inner-corona balance. While the balance is still very high, the change in distributions to a more realistic one prevents role based routing from guaranteeing perfect balance. The loss of balance increases with radius ($r = -0.91$, $p = 0.012$) but a higher density increases the balance ($r = 0.981$, $p = 0.0054$). The results for the max/mean ratio, shown in Fig. 6.2, confirm the loss of balance although this metric shows no statistically significant relationship with radius ($p = 0.126$) or density ($p = 0.213$).

Although CROBAR retains high balance in the uniform distribution there is one serious consequence which is the loss of connectivity as shown in Fig. 6.3, which shows that connectivity drops from 100% to between 90.2%% ($\pm 3.5\%$) and 82.1% ($\pm 8.6\%$). As with the balance, the connectivity falls with radius ($r = -0.953$,

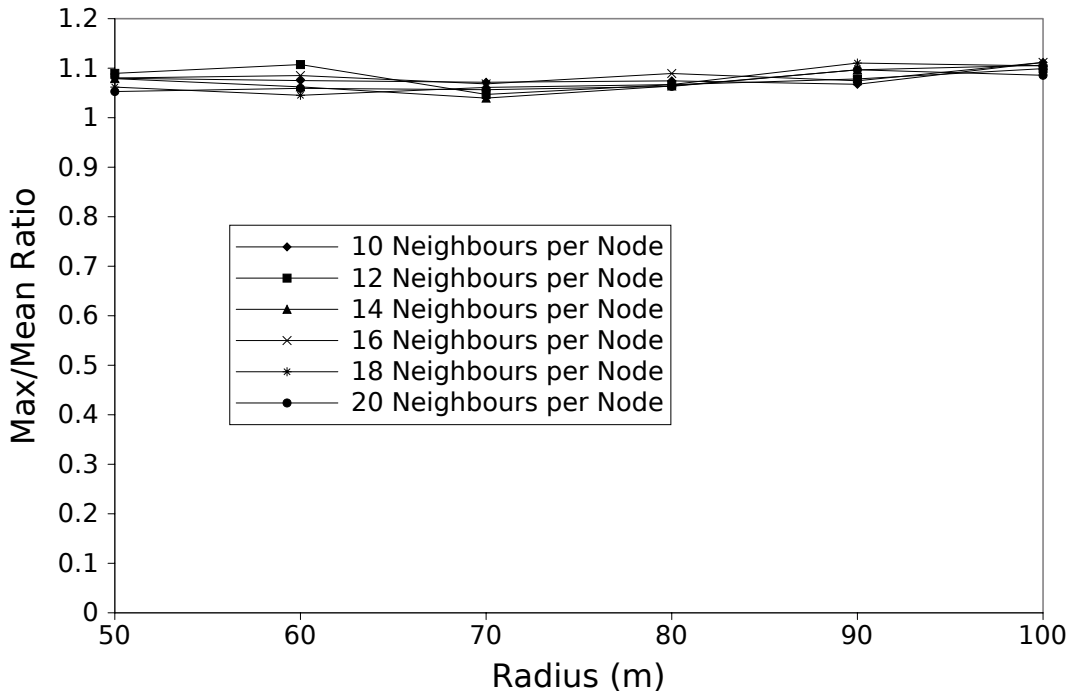


Figure 6.2: The results for the max/mean ratio are similar to those of balance, showing that role based routing can only guarantee perfect balance in unrealistic circumstances.

$p = 0.0032$) but increases with density ($r = 0.922$, $p = 0.00895$).

The reason for this behaviour is that in the uniform distribution the roles are no longer correct. The triples in level one of the routing tree cannot be sure of adopting three children each because there may not be that many nodes in the second level. Similarly, some doubles may find that there are not enough nodes in the next level to fill their quota. This results in a loss of balance.

On the other hand, there may be situations where the opposite is true and there are more nodes in a level than in the ideal distribution. In this case some of those nodes will not be able to connect to the routing tree because the nodes in the previous level have limits on the number of children they may adopt. From these results it is clear that the loss of balance is not as serious as the loss of connectivity.

The difference between the uniform and perfect distributions is very small, as shown in Table 6.1 which shows the average difference in the number of nodes per level between the uniform and perfect distributions during the simulations. This

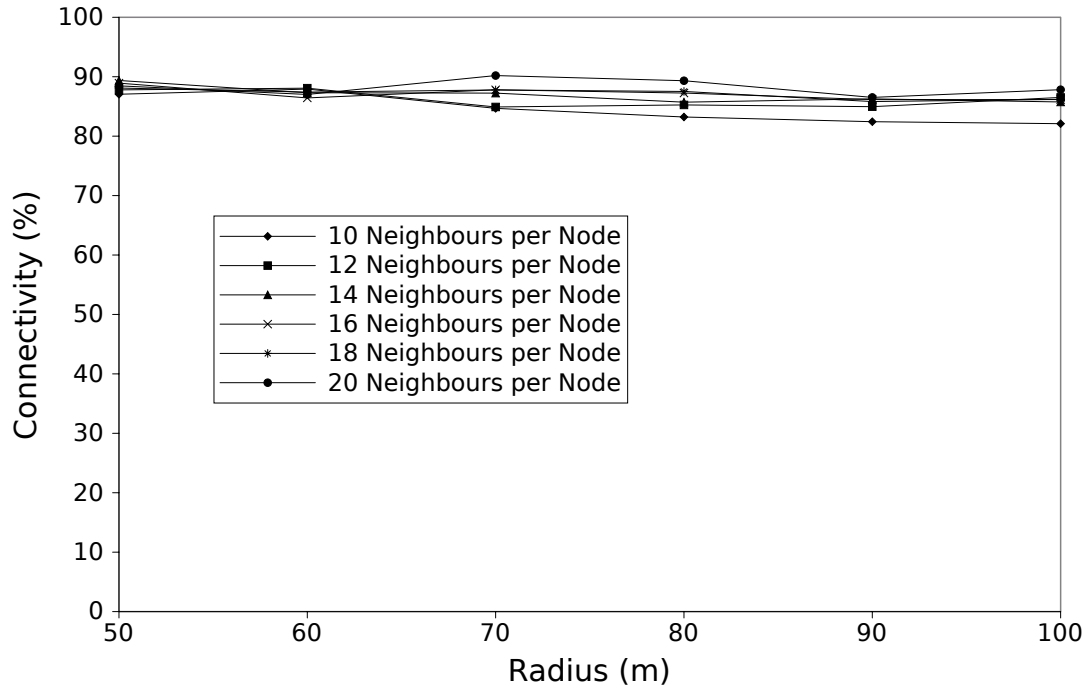


Figure 6.3: More serious than the small loss in balance is the larger loss in connectivity.

Radius	10 Neighbours	12 Neighbours	14 Neighbours	16 Neighbours	18 Neighbours	20 Neighbours
50m	0.00671	0.00594	0.0107	0.01154	0.01214	0.01593
60m	0.01646	0.01455	0.00799	0.01738	0.01443	0.01526
70m	0.01736	0.01886	0.02224	0.01722	0.01725	0.01439
80m	0.01588	0.01611	0.01642	0.01154	0.01065	0.01444
90m	0.01633	0.00884	0.01244	0.01305	0.01131	0.00859
100m	0.01954	0.02141	0.01832	0.01958	0.01231	0.01656

Table 6.1: The average difference between the uniform and perfect distributions

small change, though, prevents the role based routing approach from achieving perfect inner-corona balance.

6.3 Distributed Implementation

Although role based routing is sensitive to the ideal circumstances it nevertheless can guarantee perfect inner-corona balance which is something the existing distributed algorithms cannot do. Therefore, in more realistic scenarios it might be expected that role based routing can produce higher balance than the benchmark MHS.

In this section two distributed versions of role based routing are considered. The first, called ROBAR (ROle BAsed Routing) follows the rules identified in Section 6.1 perfectly which, as shown above, can result in decreased connectivity. The second, called ROBAR-FC (ROBAR Fully Connected), adds another rule that states that once all nodes have filled their quotes (or adopted as many children as they are able to) then nodes may adopt more children if some have been left unconnected to the routing tree. This relaxation of the roles should provide higher connectivity.

6.3.1 ROBAR

The ROBAR algorithm begins at the sink node which sets its level to zero and has no limitations on the number of children it may adopt. The routing tree is built up level by level in rounds such that one new level of the tree is added per round. The nodes who join the tree as children in one round act as the parents in the next round.

Each round consists of four steps: advertising, requesting adoption, confirming adoption and appointing doubles. During the advertising step the parents broadcast advert packets, **ADV**, that include their ID, location, hop count from the sink, hc, and their role, ie triple, double or single. Any node that is not yet in the tree that receives an **ADV** packet is a child node for that round. Child nodes store the information from all the beacons they receive during the advertising step in a table of potential parents. After transmitting an **ADV** packet each parent waits for a predetermined time, t_{req}, during which it gathers adoption requests to its advert from the child nodes. The children, meanwhile wait a predetermined time, t_{adv}, after receiving their first advert packet during which they collect **ADV** packets.

In the requesting adoption step, which each child node starts after time t_{adv} from the time it received its first **ADV** packet, the child nodes go through their list of potential parents and select the parent which is closest to the sink as shown in Algorithm 6.2. This is their ideal parent and they transmit an adoption request packet, **REQ**, to it which includes the number of potential parents the child node has, n_p. The parents store the information received in these adoption request packets in a table of potential children.

Algorithm 6.2 ChooseBestParent

```
1: function ChooseBestParent(potentialParents)
2:     if potentialParents.size() > 0 then
3:         chosenParent = NULL
4:         furthestDistance = 0
5:         for all parent ∈ potentialParents do
6:             if distance(parent.location,my.location) > furthestDistance then
7:                 furthestDistance = distance(parent.location,my.location)
8:                 chosenParent = parent
9:             end if
10:        end for
11:        n_p = potentialParents.size()
12:        send REQ(my.ID,my.location,n_p) → chosenParent
13:        potentialParents.remove(chosenParent)
14:    end if
15: end function
```

The next step is the confirming adoption stage which each parent starts after time t_{req} from the time it transmitted its **ADV** packet, during which the parent nodes select the top q ideal children from the list of potentials where q is the parent's quota (i.e. three for triples, two for doubles and one for singles). The chosen children are the ones with the fewest potential parents because if this parent does not adopt them they may be unable to be adopted by another whereas a child with more options is more likely to be adopted by another parent if not adopted by this one. If two or more children have the same number of potential parents then the one which is furthest from the parent is chosen as described in Algorithm 6.3. The parent broadcasts an adoption confirmation packet, **ADPT**, which is received by all the child nodes in range and serves both to confirm the child-parent relationship with the chosen children and also to allow other children to update their list of potential parents. The **ADPT** packet contains a field, *space*, which specifies how many more children the parent can adopt. The child nodes keep track of these adoption confirmation packets and if they receive one in which $space == 0$ they can remove that parent from their list of potential parents since it will be unable to adopt them now.

The second and third steps should repeat until all nodes have been adopted, all parents have filled their quotas, or all potential parent lists are empty. The process must also be synchronised so that all the nodes that were children in one round (excluding those that did not connect to the tree) become parents in the

Algorithm 6.3 Choose Best Child

```
1: function chooseBestChild(potentialChildren,children)
2:     chosenChildren = {}
3:     for i ← 1, (my.maxChildren - children.size() do
4:         fewestOptions = ∞
5:         furthestDist = 0
6:         bestChild = NULL
7:         for all child ∈ potentialChildren do
8:             if child.n_p < fewestOptions then
9:                 fewestOptions = child.n_p
10:                bestChild = child
11:                furthestDist = dist(child.location,my.location)
12:            else if child.n_p == fewestOptions then
13:                if dist(child.location,my.location) > furthestDist then
14:                    furthestDist = dist(child.location,my.location)
15:                    bestChild = child
16:                end if
17:            end if
18:        end for
19:        if bestChild != NULL then
20:            chosenChildren.add(bestChild)
21:            potentialChildren.remove(bestChild)
22:        end if
23:    end for
24:    space = my.maxChildren - (children.size() + chosenChildren.size())
25:    broadcast ADPT(my.ID,chosenChildren,space)
26:    children.add(chosenChildren)
27: end function
```

next round at approximately the same time and broadcast beacons at about the same time. If this does not happen then child nodes will start receiving adverts from some parents much earlier than from others so that they do not generate a complete list of potential parents before starting the adoption request stage.

To ensure that the rounds are synchronised but are not too short as to prevent a child node from being adopted, each round has enough request-confirmation cycles to allow each child node to send an advert to every one of its potential parents. Since this figure is different for different children a higher upper limit is chosen that can be known by the nodes without knowing the full topology of the network, namely the number of neighbours. The sink can easily identify the approximate number of neighbours in the network because all of its neighbours become its children and the nodes are uniformly distributed meaning that all nodes have approximately the same number of neighbours. Since not all neighbours are potential parents this value serves as an upper bound on the number of potential parents a child node would have. The sink can include this figure in its adoption-confirmation packet and it can then be flooded through the network as part of the advert packets so that every node is aware of how many adoption request and confirmation cycles to wait for before the next round starts.

The parent nodes wait for some predetermined time, t_{round}, from the time they transmit their **ADV** packets after which they start the final round in which they appoint doubles from their children. Since the parents do not know the topology information they cannot know which of their children is more suited to being a double they therefore simply select one (or two in the case of triples) at random, as shown in algorithm 6.4, and broadcast an appointment packet **APPT** which includes a list of new doubles. The **APPT** packet also serves as an instruction to start the next round and all child nodes that receive it become parent nodes and broadcast their own **ADV** packets (using a random back-off to prevent collisions) thereby beginning the next round.

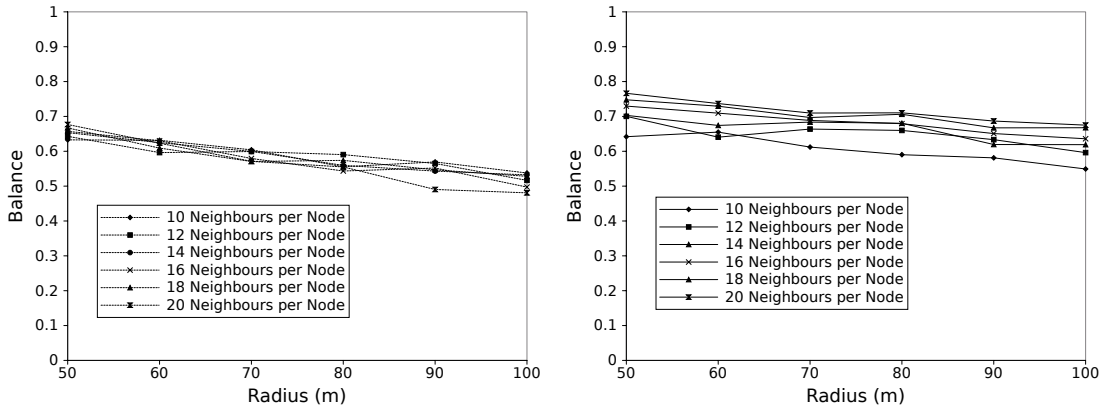
The ROBAR algorithm was simulated in a number of network configurations with radius ranging from 50m to 100m and density ranging from ten neighbours per node to twenty. The nodes were distributed using a random uniform distribution and nodes could only directly communicate with other nodes within 10m of themselves. ROBAR was compared against the benchmark MHS algorithm. Fig. 6.4 shows the results for the balance metric which show that ROBAR produced

Algorithm 6.4 Appoint Doubles

```
1: function appointDoubles(children)
2:     if my.role == double || my.role == triple then
3:         chosenDoubles = {}
4:         index = random(0,children.size())
5:         chosenChild = children.get(index)
6:         chosenDoubles.add(chosenChild)
7:         if my.role == triple then
8:             remainingChildren = children
9:             remainingChildren.remove(chosenChild)
10:            index = random(0,remainingChildren)
11:            chosenChild = remainingChildren.get(index)
12:            chosenDoubles.add(chosenChild)
13:        end ifbroadcast APPT(my.ID,chosenDoubles)
14:    end if
15: end function
```

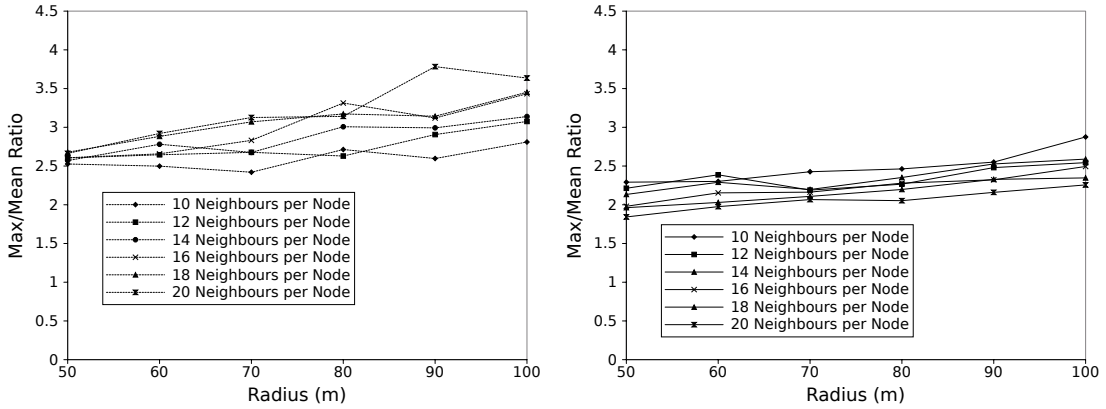
higher balance than MHS in all configurations. The improvement ranged from 1.3% (±5.65%) to 40.4% (±10.02%) and increased with both radius ($r = 0.962$, $p = 0.0022$) and density ($r = 0.985$, $p = 0.0003$). Similar results were seen with the max/mean ratio shown in Fig. 6.5. In two cases MHS actually resulted in a lower ratio than ROBAR by 2.33% (±10.05%) and 0.26% (±6.48%) but in all other configurations ROBAR had a lower ratio up to 42.91% lower (±4.57%). The amount of improvement shows no statistically significant correlation with radius ($r = 0.52$, $p = 0.291$) but increases with density ($r = 0.9899$, $p = 0.00015$). In the best case ROBAR has a max/mean ratio of 2.16 (±0.22) compared to MHS's 3.78 (±0.43) which corresponds to a network lifetime increase of 75%.

However, in order to achieve these improved levels of balance ROBAR trades-off connectivity as shown in Fig. 6.6. While MHS always results in all nodes connecting to the routing tree, the strict adherence to roles under ROBAR means that in the best case in my experiments only 92% (±1.6%) of nodes can connect to the tree and in the worst case only 46% (±4.3%) can. As is evident from the graph, the connectivity falls with radius ($r = -0.998$, $p = 8.32 \times 10^{-6}$) but increases with density ($r = 0.975$, $p = 0.0009$).



(a) MHS (b) ROBAR

Figure 6.4: ROBAR consistently produced greater balance than MHS and the improvement increased with both radius and density.



(a) MHS (b) ROBAR

Figure 6.5: The max/mean ratio under ROBAR is almost always lower than under MHS which, in the best case, corresponds to a 75% increase in lifetime.

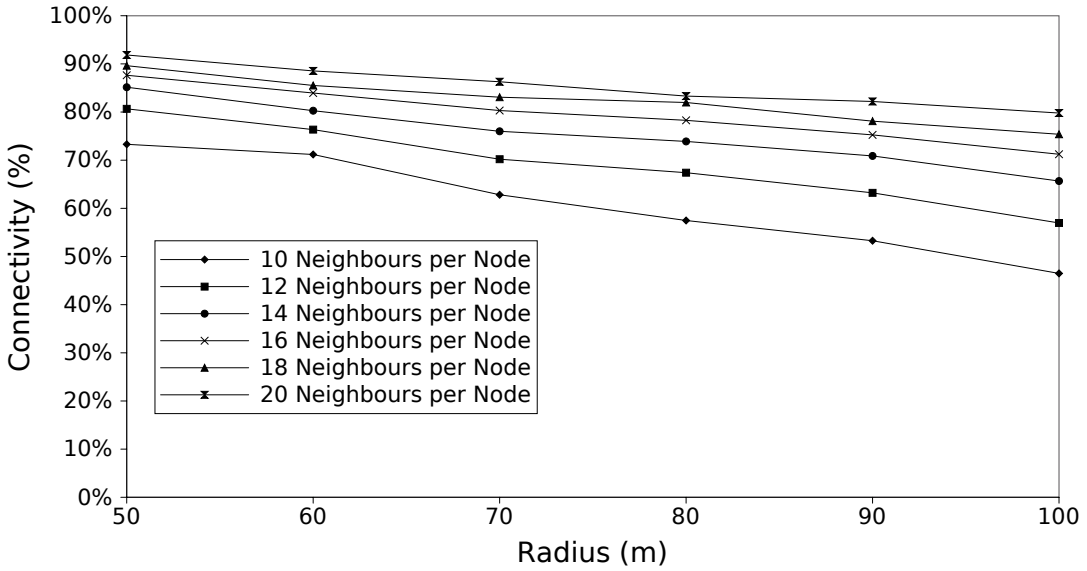


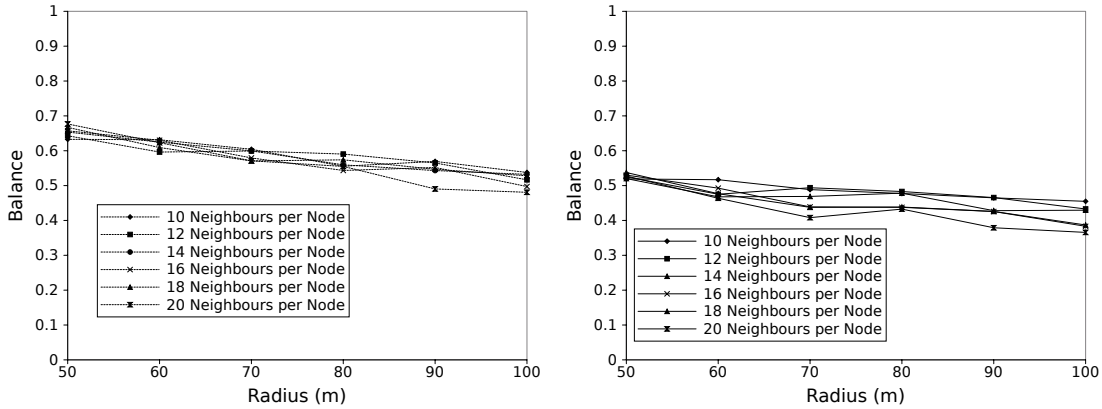
Figure 6.6: Strict adherence to the roles under ROBAR means that many nodes are unable to connect to the routing tree.

6.3.2 ROBAR-FC

The results in the previous section showed that role based routing can significantly increase the balance and lifetime of sensor networks but in order to achieve this improvement redundant nodes must be added to the network because not all nodes are able to connect to the sink. In this section a modified role based routing protocol, ROBAR-FC (ROBAR-Fully Connected) is described and simulated.

ROBAR-FC initially behaves exactly like the original ROBAR algorithm and nodes adhere strictly to their roles. However, at the end of each round, before parents appoint doubles, the first three steps are repeated once more but this time the parent nodes ignore their roles and adopt as many children as request adoption. Every node will therefore be able to connect to the routing tree but because the roles are ignored for part of the tree construction process the balance is likely to fall.

The simulation results confirm that connectivity under ROBAR-FC is 100% but that this comes at a cost. As Fig. 6.7 shows, the effect of modifying ROBAR to achieve full connectivity is that the balance it can achieve falls below that of the benchmark MHS. In fact, MHS now has between 16.4% ($\pm$7.3%) and 39.9%



(a) MHS (b) ROBAR-FC

Figure 6.7: By modifying ROBAR to allow full connectivity, the levels of inner-corona balance fall significantly and are lower than the benchmark.

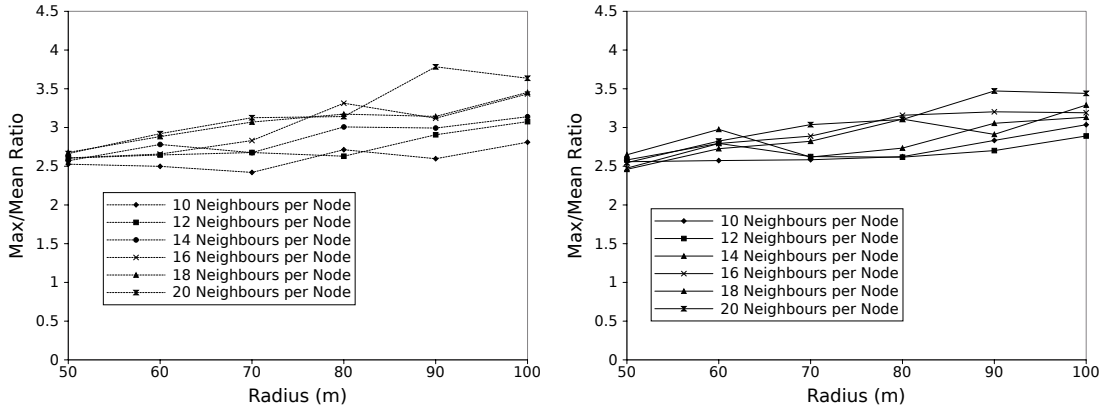
($\pm 5.22\%$) higher balance and this difference increases with density ($r = 0.993$, $p = 8.14 \times 10^{-5}$) but is invariant with radius ($p = 0.936$).

However, the results for the max/mean ratio give a different picture as seen in Fig. 6.8. This measure shows that although at low densities ROBAR-FC performs worse than the benchmark, at higher densities it starts to perform better. Averaging across all radius values MHS has a max/mean ratio that is 3.80% ($\pm 4.68\%$) lower than ROBAR-FC when there are ten neighbours per node but at 20 neighbours its max/mean ratio is 4.47% ($\pm 2.81\%$) higher. There is a positive correlation between the difference in max/mean ratios and density ($r = 0.843$,$p = 0.035$). Based on these results it is difficult to conclude that ROBAR-FC is a better approach in terms of load balancing and lifetime than the benchmark MHS.

6.4 Chapter Summary and Conclusions

In this chapter a new approach to maximising network lifetime, namely constructing a routing tree designed to maximise inner-corona balance in a distributed fashion, has been explored. The proposed protocol, role based routing, is the first attempt to maximise network lifetime in this way.

In the previous chapter it was proven that a strategy focusing on maximising degree balance (used in protocols such as MBT [HCWC09] and MHS [CZYG10])



(a) MHS (b) ROBAR-FC

Figure 6.8: The relationship between the benchmark and ROBAR-FC in terms of the max/mean ratio is unclear. It cannot be claimed with certainty that ROBAR-FC outperform MHS on this measure although the data suggests that it might.

cannot guarantee perfect inner-corona balance even in ideal circumstances. However, role based routing is able to guarantee this which suggests that it should provide higher balance in more realistic scenarios. Indeed, results show that it does outperform the benchmark MHS protocol leading to a lifetime increase of up to 75%. There is a major cost, though, to this improved balance which is that many nodes are unable to connect to the routing tree. In order to use the ROBAR protocol in practice, the network would need to contain a significant percentage of entirely redundant nodes so that, despite the low connectivity, the number of nodes connected to the routing tree would be enough to meet the application requirements.

Although modifying the original ROBAR algorithm to provide full connectivity (ROBAR-FC) is possible, it results in a loss of balance and it is impossible to claim with any confidence that the lifetime under ROBAR-FC is any higher than under MHS. From this chapter it is safe to conclude that although role based routing seemed like a promising approach to distributed inner-corona balance, the loss of connectivity is too high to be acceptable in most cases.

Chapter 7

Degree Constrained Routing

In the previous chapter a novel approach, role based routing (ROBAR), to inner-corona balancing was proposed and analysed and it was found that improved balance and extended lifetime were achievable if connectivity was sacrificed. The motivation behind the approach was that if a method can guarantee perfect inner-corona balance in ideal circumstances then it is likely to have higher balance in more realistic scenarios than an approach that cannot make that guarantee. However, role based routing was extremely sensitive to the ideal circumstances such that even moving from a perfect to a uniform distribution resulted in a loss of perfect balance.

In this chapter another approach is considered which makes use of the trade-off between balance and connectivity, namely degree constrained routing (DECOR). In a similar way to ROBAR, DECOR imposes a limit on the number of children that nodes can adopt. However, while ROBAR had different limits for different nodes based on their role in the network, the DECOR algorithm applies limits to nodes based on their position in the routing tree.

In the next section the theory underpinning DECOR is explained and initial simulations are described that prove that DECOR can guarantee perfect balance in ideal circumstances by sacrificing connectivity. In Section 7.2 the DECOR approach is applied to a distributed routing protocol which is tested, showing that it can achieve high inner-corona balance but at the cost of connectivity and latency. Finally, this initial approach is extended in Section 7.3 with a second phase that can be used to improve the connectivity and latency of the initial

solution.

7.1 Theory Behind Degree Constrained Routing

Degree constrained routing arises from the observation that inner-corona imbalance occurs because of the way in which the number of nodes per corona changes and the desire to have all the nodes connect to the routing tree. If the number of nodes per corona remained constant then it would be simple to create a balanced tree - all that is needed is for every parent to adopt exactly one child. However, as previously stated several times, Macedo analysed the growth in the number of nodes per corona in uniform networks [Mac09] and found that the average number of nodes per parent in corona c_i, C_i was:

$$C_i = \frac{2i+1}{2i-1} \tag{7.1}$$

Because the number of children per parent is always a fraction (except for parents in the first level of the tree), in order to have all nodes connect to the tree some nodes must adopt more children than others which causes imbalance. For a distributed routing algorithm that deals with the assignment of parents to children for each level independently of every other level, there is a clear trade-off between balance and connectivity. Degree constrained routing prioritises balance over connectivity by making parents adopt the same number of children and leaving the surplus disconnected from the tree.

Barring routing and relay holes, degree constrained routing can be achieved by placing an upper limit on the number of children each parent can adopt in a similar way to role based routing. However, some care is required that the limit is low enough to allow all parents to fill their quota because otherwise there is still imbalance. A simple choice for a quota would be to use $\lfloor C_i \rfloor$ because this value is guaranteed to be a whole number (by definition) and there will be enough child nodes to allow all parents to fill their quota. However, this limit does not take into consideration the actual growth in the number of nodes per corona which is shown in Table 7.1. If the number of nodes in the inner-most corona is n and the

quota is set to $\lfloor C_i \rfloor$, then only $3n$ nodes can be adopted in every corona despite the fact that the number of nodes is increasing. For example, in the tenth corona there would be $19n$ nodes and only $3n$ of those would be able to connect to the tree, leaving over 80% of those nodes unconnected.

In order to reduce the loss of connectivity resulting from DECOR, the quota should increase above $\lfloor C_i \rfloor$ wherever it is possible to do so and still have enough child nodes for all parents to fill their quotas. The quotas must be kept low in order to allow them to be increased frequently because the number of nodes per corona grows slowly. Since $\lfloor C_i \rfloor = 1$ for every level except the first (where it equals three) the default quota is that nodes may only adopt one child. To allow for the most frequent relaxations, the relaxed quota should be two.

For the nodes in the first level, $\lfloor C_i \rfloor = 3$ and therefore there is choice to be made as to whether to set the quota for those nodes also to three or to reduce it to two. The choice will determine which other levels can have relaxed quotas by altering the number of connected nodes in each level. Table 7.1 illustrates the impact of the two options, listing which levels can have relaxed quotas as the result of the quota of the level one nodes and what effect that has on the number of connected and disconnected nodes in each corona. If the quota for the first level is set to three then quota relaxations can only happen in levels {3,6,12,24...} whereas if the quota for level one is also two then relaxations can happen in levels {2,4,8,16...}. The results in Table 7.1 demonstrate that in some cases one choice leads to higher connectivity and in other cases the other is better, depending on the size of the network. However, since the nodes cannot know how large the network is they must choose to follow one option "blind" and so the first option (level one quota set to three) is chosen because, over the range of radius values considered in the simulations, this choice more often leads to higher connectivity (five times) than the other (twice).

The connectivity expected from degree constrained routing is significantly below 100%, falling to a minimum of 67.19%. This suggests that there may exist a trade-off involved in choosing a load balancing method. If full connectivity is desired then MHS may be chosen whereas if a lower connectivity is acceptable, degree constrained routing could be used to increase balance. The first option provides more readings in the physical space but for a shorter time. However, the aim is to modify degree constrained routing to provide high connectivity (ideally

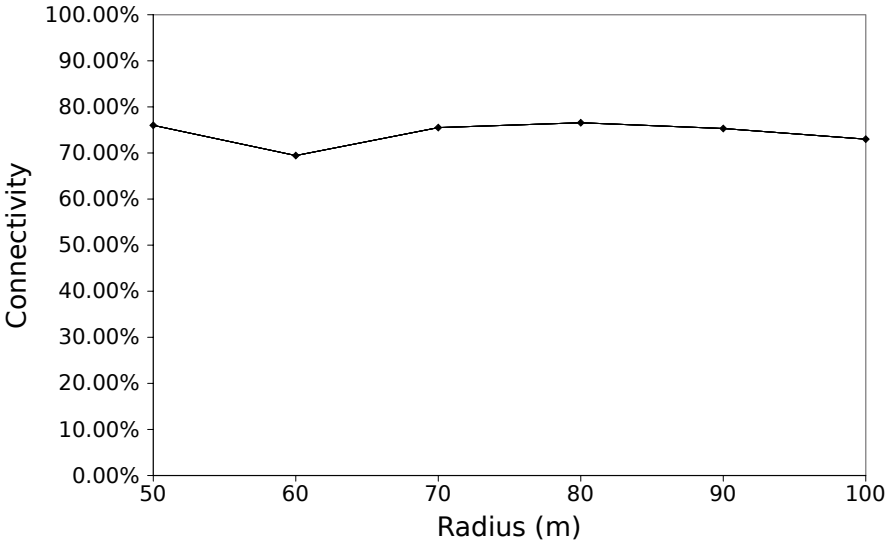


Figure 7.1: The cost of perfect inner-corona balance is reduced connectivity which varies between 76.56% and 69.44% with different radii but is independent of density.

100%) without sacrificing too much of the balance and this is discussed more in Section 7.3.

To test this theory, a centralised algorithm was developed that implemented degree constrained routing (CDECOR) in the idealised scenario described in Section 5.2.1 as shown in algorithm 7.1.

Using the same configurations as in previous simulations the first option (level one quota of three) was simulated in ideal circumstances. As expected the balance and max/mean ratios were perfect and the connectivity, shown in Fig. 7.1, was precisely as predicted by the analysis in Table 7.1.

The above results validate the theory underpinning DECOR and prove that it can guarantee perfect balance in ideal circumstances. However, it is important to know how sensitive the approach is to the perfect circumstances and this can be gauged, to some extent, by considering the same algorithm with a uniform distribution of nodes rather than a perfect one. Before testing the approach, however, the quota for the level one nodes is changed to two and the alternative approach discussed above is used (whereby nodes in levels $\{2,4,8,16\ldots\}$ have a quota of two). Since the nodes will now be distributed randomly there is no guarantee that there will be enough nodes in the second corona to allow all

Table 7.1: The effect of different quotas for level one nodes

Corona Number	Number of Nodes	Quota for level 1 nodes = 3				Quota for level 1 nodes = 2			
		Quota	Connected in Corona	Disconnected in Corona	**Connectivity**	Quota	Connected in Corona	Disconnected in Corona	**Connectivity**
1	n	3	n	0	**100%**	2	n	0	**100%**
2	$3n$	1	$3n$	0	**100%**	2	$2n$	$1n$	**75.00%**
3	$5n$	2	$3n$	$2n$	**77.78%**	1	$4n$	$1n$	**77.78%**
4	$7n$	1	$6n$	$1n$	**81.25%**	2	$4n$	$3n$	**68.75%**
5	$9n$	1	$6n$	$3n$	**76.00%**	1	$8n$	$1n$	**76.00%**
6	$11n$	2	$6n$	$5n$	**69.44%**	1	$8n$	$3n$	**75.00%**
7	$13n$	1	$12n$	$1n$	**75.51%**	1	$8n$	$5n$	**71.43%**
8	$15n$	1	$12n$	$3n$	**76.56%**	2	$8n$	$7n$	**67.19%**
9	$17n$	1	$12n$	$5n$	**75.31%**	1	$16n$	$1n$	**72.84%**
10	$19n$	1	$12n$	$7n$	**73.00%**	1	$16n$	$3n$	**75.00%**

Algorithm 7.1 CDECOR

```
1: function CDECOR(nodes,sink)
2:     for all node ∈ nodes do                          ▷ Initialise the nodes
3:         node.parent = NULL
4:         node.sinkDist = √((node.X − sink.X)² + (node.Y − sink.Y)²)
5:         node.level = 1 + ⌊node.sinkDist/transmissionRange⌋
6:         node.maxChildren = 1
7:         node.connected = false
8:         levels[node.level].put(node)
9:         if node.level == 1 then
10:            node.maxChildren = 3
11:            node.setParent = sink
12:        else if node.level ∈ {3, 6, 12, 24 . . . } then node.maxChildren = 2
13:        end if
14:    end for
15:    for all level ∈ levels do                        ▷ Assign children to parents
16:        for i ← 1, 3 do
17:            for all parent ∈ level do
18:                if parent.connected == true then
19:                    while parent.numberChildren < parent.maxChildren do
20:                        for all child ∈ levels[level+1] do
21:                            if child.parent==NULL then
22:                                child.parent = parent
23:                                child.connected = true
24:                                parent.numberChildren++
25:                            end if
26:                        end for
27:                    end while
28:                end if
29:            end for
30:        end for
31:    end for
32: end function
```

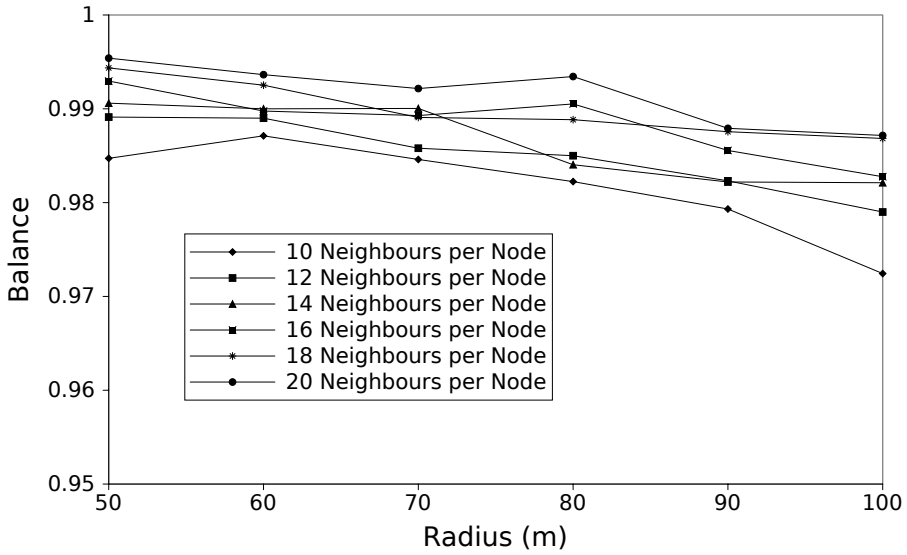


Figure 7.2: In the non-ideal scenario of uniform distribution, the balance falls slightly but still remains very high with the worst case balance being 0.98.

the nodes in the first corona to adopt three children each. If some of the first level nodes fail to fill their quota this will have a serious impact on inner-corona balance. This risk is greatly reduced by reducing the quota for the first level nodes down to two.

The results shown in Fig. 7.2 are that balance falls slightly because of the move from a perfect to a uniform distribution, although the lowest it falls to is 0.98 (± 0.011) which is very high. As is seen from the graph, balance falls with radius ($r = -0.98$, $p = 0.00055$) but increases with density ($r = 0.99$, $p = 0.00016$). The results for the max/mean ratio (Fig. 7.3) underline the high level of balance showing that in the worst case the ratio is 1.084 (± 0.07) which corresponds to a reduction in lifetime of only 8.4% from the idealised scenario. As with balance, the max/mean ratio gets worse (i.e. increases) with radius ($r = 0.898$, $p = 0.015$) but improves (i.e. falls) with density ($r = -0.969$, $p = 0.00146$)

Fig. 7.4 shows the connectivity with the uniform distribution which varies between 60.97% ($\pm 4.69\%$) and 73.11% ($\pm 5.12\%$). As with the perfect distribution, the connectivity varies with radius but in a non-linear fashion. There is some improvement with density ($r = 0.858$, $p = 0.0288$). Compared to the perfect distribution, there is a loss of connectivity of up to 15.58 (± 5.98) percentage

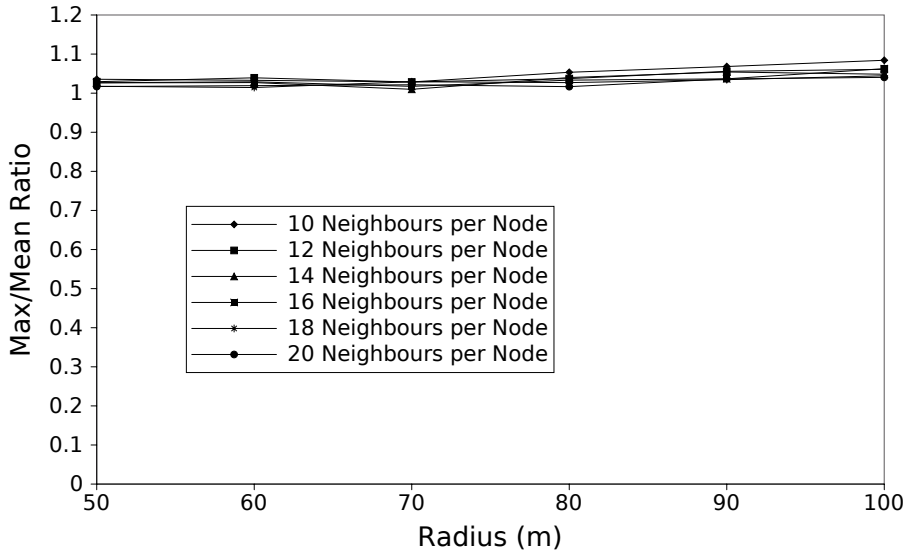


Figure 7.3: The max/mean ratio increases when a uniform distribution is used but remains low and the network lifetime is never reduced by more than 8.4%.

points although the average across all configurations was 6.62 ($\pm 5.61\%$) percentage points. Here too there is some reduction in the difference with density ($r = -0.86$, $p = 0.0279$).

These results indicate that the degree constrained approach is less reliant on the idealised circumstances than the role based one was which suggests that a distributed implementation may perform better than ROBAR and also better than the benchmark, MHS.

7.2 Distributed Degree Constrained Routing

The results in the previous section suggest that degree constrained routing is a useful approach for achieving inner-corona balance. In ideal circumstances the approach can guarantee perfect balance and the balance remains very high with a uniform distribution of the nodes. The trade-off for balance is a reduction in connectivity but even in the uniform distribution more than two thirds of the nodes are able to connect to the routing tree.

In this section the degree constrained approach is implemented as a distributed

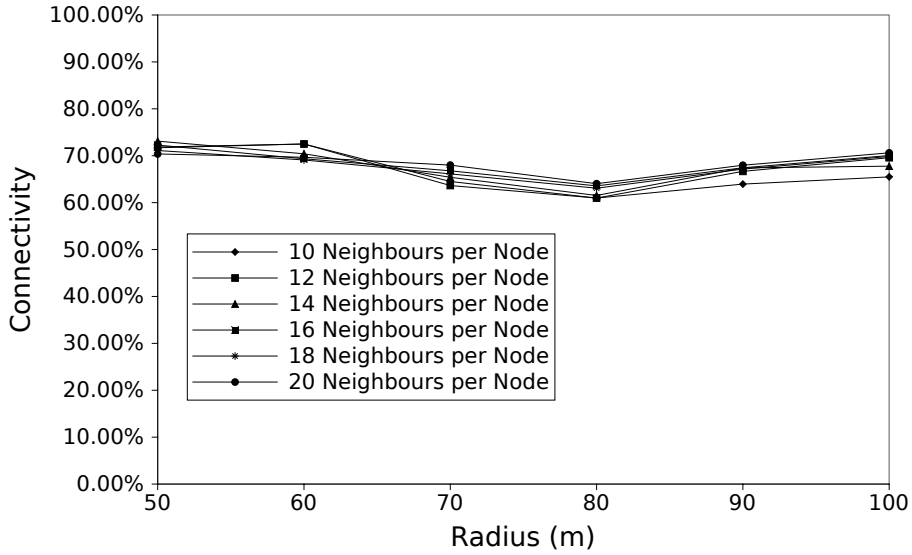


Figure 7.4: Connectivity falls using a uniform distribution by an overall average of 6.62 percentage points compared to the perfect distribution.

routing protocol, DECOR, and tested in the more realistic scenario of a uniform distribution with a maximum transmission range of 10m per node. The distributed version uses the level one quota of two children per node and therefore has increased quotas of two children per node in levels {2,4,8,16...} of the routing tree.

The DECOR algorithm is very similar to the ROBAR one. It begins at the sink node which sets its level to zero and has no limitations on the number of children it may adopt. The routing tree is built up level by level in rounds such that one new level of the tree is added per round. The nodes who join the tree as children in one round act as the parents in the next round.

Each round consists of three steps: advertising, requesting adoption and confirming adoption. During the advertising step the parents broadcast advert packets, **ADV**, that include their ID, location and hop count from the sink, *hc*. The **ADV** packet also contains the parent's quota q whereas with ROBAR it contained the parent's role. Any node that is not yet in the tree that receives an **ADV** packet is a child node for that round. Child nodes store the information from all the adverts they receive during the advertising step in a table of potential parents. After transmitting an **ADV** packet each parent waits for time t_{req} during which it gathers adoption requests from the child nodes. The children, meanwhile wait

Algorithm 7.2 Choose Best Parent

```
1: function chooseBestParent(potentialParents)
2:     if potentialParents.size() > 0 then
3:         chosenParent = NULL
4:         furthestDistance = 0
5:         for all parent ∈ potentialParents do
6:             if distance(parent.location,my.location) > furthestDistance then
7:                 furthestDistance = distance(parent.location,my.location)
8:                 chosenParent = parent
9:             end if
10:        end for
11:        n_p = potentialParents.size()
12:        send REQ(my.ID,my.location,n_p) → chosenParent
13:        potentialParents.remove(chosenParent)
14:    end if
15: end function
```

time t_{adv} after receiving their first advert packet during which they collect **ADV** packets.

In the requesting adoption step, which each child node starts after time t_{adv} from the time it received its first **ADV** packet, the child nodes go through their list of potential parents and select the parent which is closest to the sink as shown in Algorithm 7.2. This is their ideal parent and they transmit an adoption request packet, **REQ**, to it which includes the number of potential parents the child node has, n_p. Having sent the request the child removes the chosen parent from its list of potentials to prevent it selecting this parent again. If the selected parent does not adopt the child following this request then it must have filled its quota with other nodes in which case there is no point requesting adoption from it a second time. The parents, meanwhile, store the information received in these adoption request packets in a table of potential children.

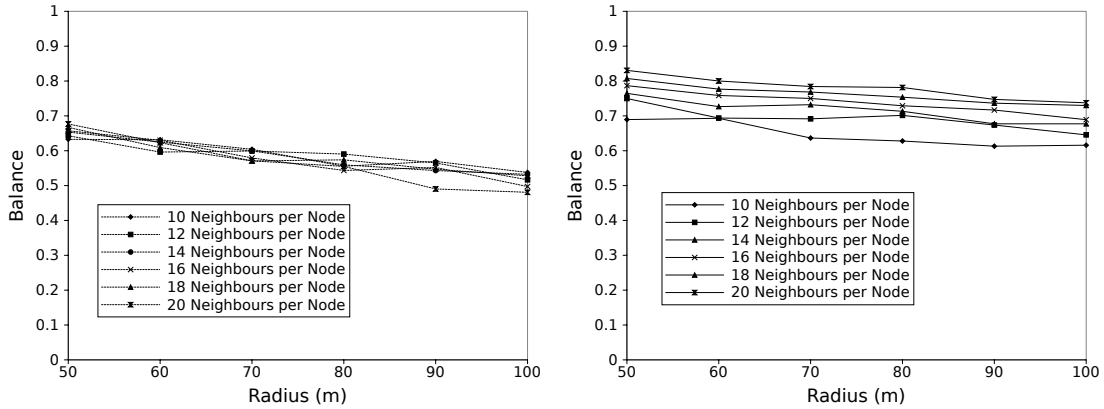
The next step is the confirming adoption stage which each parent starts after time t_{req} from the time it transmitted its **ADV** packet, during which the parent nodes select the top q ideal children from the list of potentials where q is the parent's quota. The chosen children are the ones with the fewest potential parents because if this parent does not adopt them they may be unable to be adopted by another whereas a child with more options is more likely to be adopted by another parent if not adopted by this one. If two or more children have the same number of

Algorithm 7.3 Choose Best Child

```
1: function chooseBestChild(potentialChildren,children)
2:     chosenChildren = {}
3:     for i ← 1,(my.maxChildren - children.size()) do
4:         fewestOptions = ∞
5:         furthestDist = 0
6:         bestChild = NULL
7:         for all child ∈ potentialChildren do
8:             if child.n_p < fewestOptions then
9:                 fewestOptions = child.n_p
10:                bestChild = child
11:                furthestDist = 0
12:            else if child.n_p == fewestOptions then
13:                if dist(child.location,my.location) > furthestDist then
14:                    furthestDist = dist(child.location,my.location)
15:                    bestChild = child
16:                end if
17:            end if
18:        end for
19:        if bestChild != NULL then
20:            chosenChildren.add(bestChild)
21:            potentialChildren.remove(bestChild)
22:        end if
23:    end for
24:    space = my.maxChildren - (children.size() + chosenChildren.size())
25:    broadcast ADPT(my.ID,chosenChildren,space)
26:    children.add(chosenChildren)
27: end function
```

potential parents then the one which is furthest from the parent is chosen as described in Algorithm 7.3. The parent broadcasts an adoption confirmation packet, **ADPT**, which is received by all the child nodes in range and serves both to confirm the child-parent relationship with the chosen children and also to allow other children to update their list of potential parents. The **ADPT** packet contains a field, *space*, which specifies how many more children the parent can adopt. The child nodes keep track of these adoption confirmation packets and if they receive one in which $space == 0$ they can remove that parent from their list of potential parents since it will be unable to adopt them now.

The DECOR algorithm, like ROBAR, must be synchronised but also have enough



(a) MHS (b) DECOR

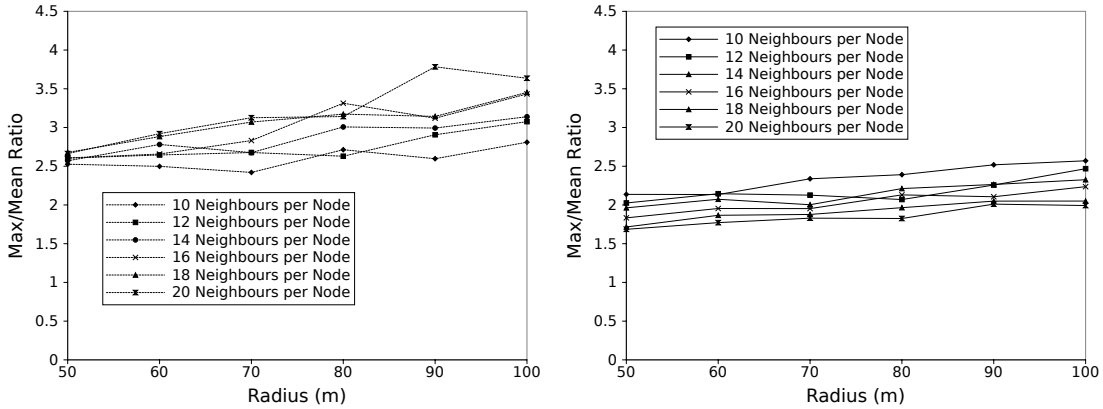
Figure 7.5: DECOR results in up to 53.41% more balance than the benchmark MHS protocol and the difference between them increases with both radius and density.

cycles of adoption request-confirmation steps to ensure that nodes are not unnecessarily prevented from joining the tree. The solution is the same as with ROBAR that the sink uses its number of children as an approximation to the number of neighbours per node throughout the network and uses this as the number of adoption request-confirmation cycles.

DECOR was simulated over a range of radii and densities and compared to the benchmark MHS algorithm. Fig. 7.5 shows that the DECOR protocol significantly outperforms the benchmark MHS protocol in terms of balance up to a maximum increase of 53.41% (±10.67%). The balance achieved by DECOR falls with radius ($r = -0.988$, $p = 0.00023$) but increases with density ($r = 0.987$, $p = 0.00025$). However, its improvement over MHS increases with both radius ($r = 0.987$, $p = 0.00025$) and density ($r = 0.99$, $p = 0.00014$).

Similar results are found for the max/mean ratio shown in Fig. 7.6 which show that the ratio is between 3.04% (±11.07%) and 46.86% (±5.24%) lower using DECOR than MHS. These differences correspond to a lifetime improvement of up to 88.17% and the improvement increases with density ($r = 0.988$, $p = 0.00021$), although it is not significantly correlated with radius ($p = 0.174$).

Fig. 7.7 shows the loss of connectivity that is traded for the improvement to balance. Connectivity varies between 87.74% (±1.92%) and 43.07% (±3.71%) with higher densities having higher connectivity ($r = 0.976$, $p = 0.00089$). Although



(a) MHS (b) DECOR

Figure 7.6: DECOR reduces the max/mean ratio by up to 46.86% which corresponds to an improvement in lifetime of up to 88.17%.

the loss of connectivity is not strictly speaking linear with radius because different radius values result in different numbers of coronas with relaxed quotas, nevertheless a linear correlation is a good approximation for the behaviour over the range of values in the simulations and it confirms that connectivity falls with radius ($r = -0.994$, $p = 6.17 \times 10^{-5}$).

7.3 DECOR Fully Connected

For other routing protocols, such as MHS, the protocol finishes with the nodes in the outer-most corona because they have no disconnected nodes within range that could be added to the routing tree. With DECOR, however, the connectivity is not 100% and so there are disconnected nodes within communication range of the nodes in the outer-most corona that could connect to those nodes. What prevents them from doing so is that they are closer to the sink than the connected nodes and greedy forwarding insists that packets move only in the direction of the sink. This observation opens up the possibility of increasing the connectivity achieved by the DECOR algorithm by relaxing the greedy forwarding requirement and allowing child nodes to connect to parents even if the parent is further away from the sink than the child node.

The likely improvement to connectivity comes at a cost though. In the first instance balance will probably fall as the number of disconnected nodes is likely

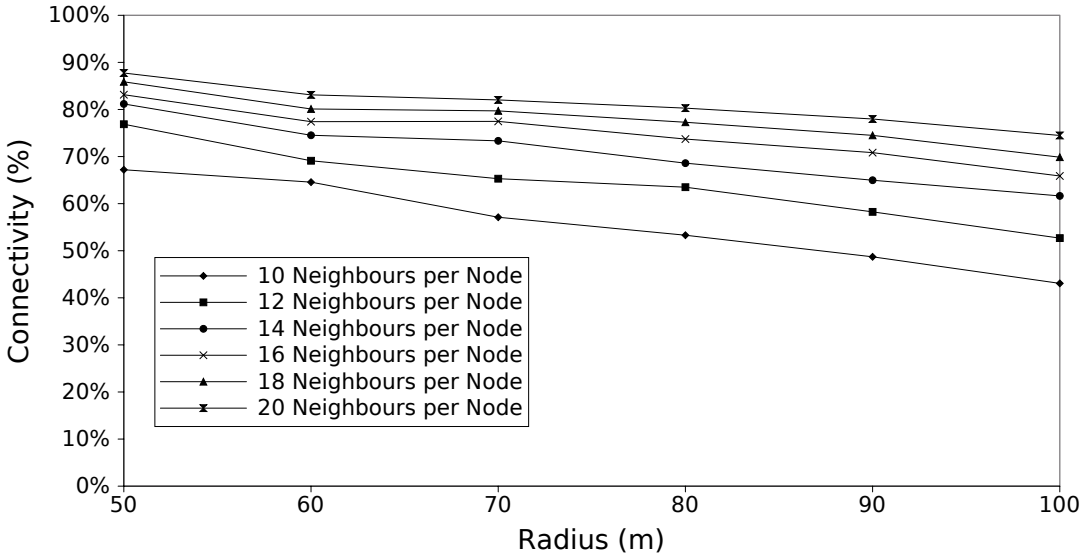


Figure 7.7: The price that DECOR pays for extra balance is a loss in connectivity. In the worst case connectivity falls to 43.07% but higher densities reduce the loss.

to be too low to allow all parents to adopt an extra child. A second cost is in terms of latency because the nodes that are able to connect to the routing tree through this method are connecting to it at a greater depth than their physical position would normally locate them. In this section, the DECOR algorithm is modified to allow nodes to adopt children that are closer to the sink than they are and then this updated version of DECOR is simulated with the same network configurations as in the previous section.

Fig. 7.8 shows that connectivity is greatly improved by relaxing the greedy forwarding requirement in the new version of DECOR. In the worst case connectivity is still greater than 90% (90.23% ±1.49%) and it increases logarithmically with density ($r = 0.93$, $p = 0.0073$) to a maximum of 99.93% (±0.063%). The cost in terms of balance from this extra connectivity, shown in Fig. 7.9, is surprisingly small, never more than 5.02% (±4.07%) lower than with greedy forwarding and decreasing logarithmically with density ($r = -0.894$, $p = 0.016$). The same results are found with the max/mean ratio, shown in Fig. 7.10, which is, on average, only 4.09% higher without greedy forwarding than with it and the corresponding increase in lifetime is up to 84.57% which compares favourably to the lifetime increase of 88.17% with the original version.

The major cost of this extra connectivity is in latency, as seen in Fig. 7.11 which

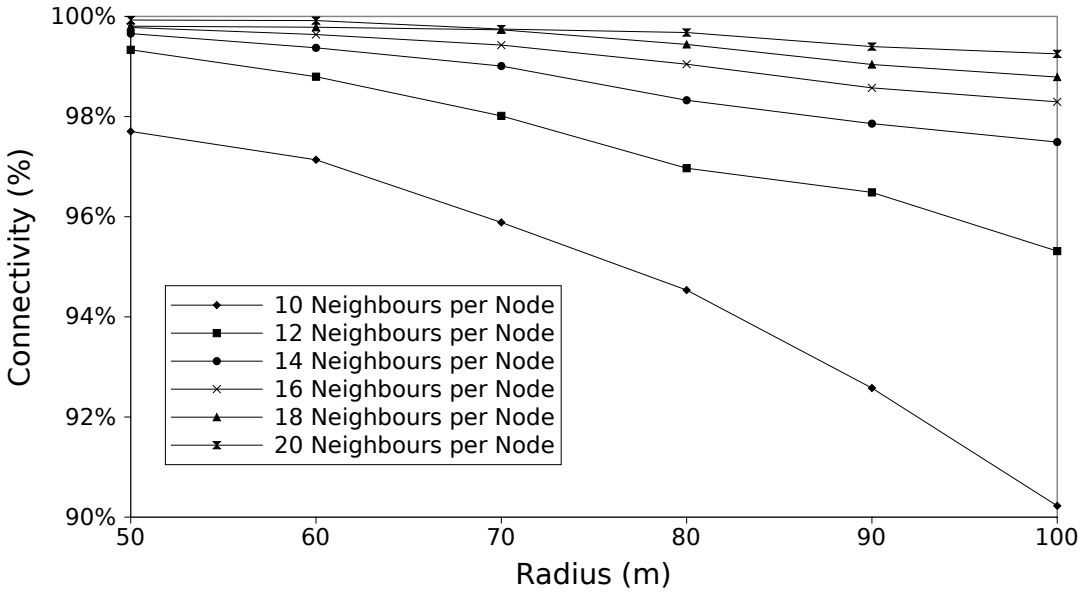
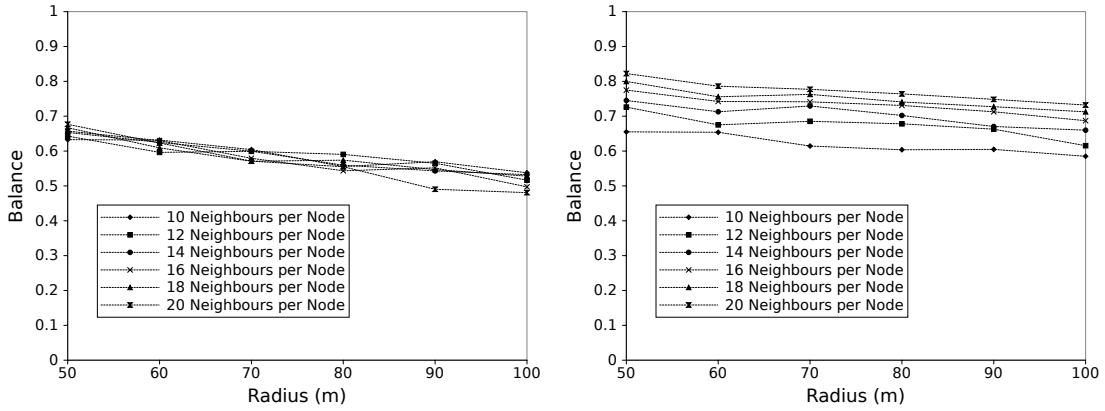
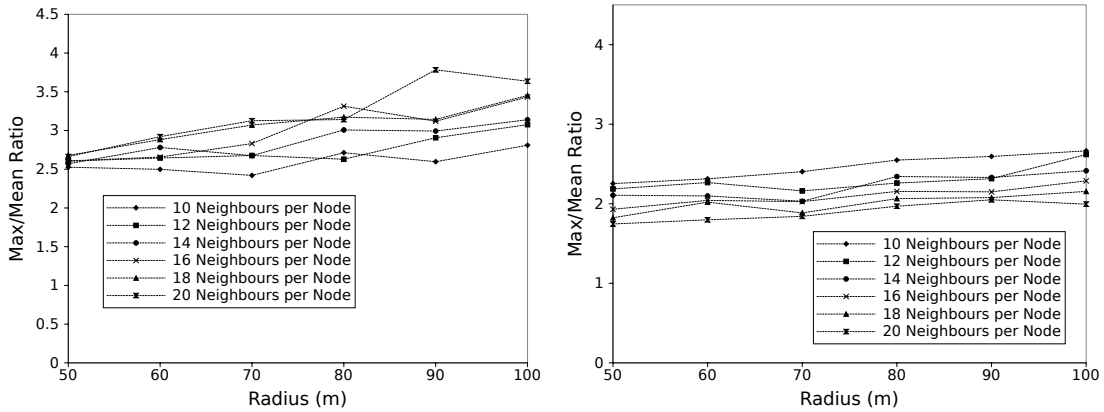


Figure 7.8: Removing the greedy forwarding limitation results in significantly higher connectivity up to 99.93%



(a) MHS (b) DECOR

Figure 7.9: The balance achieved by DECOR when greedy forwarding is relaxed remains high and similar to the balance with the restriction.



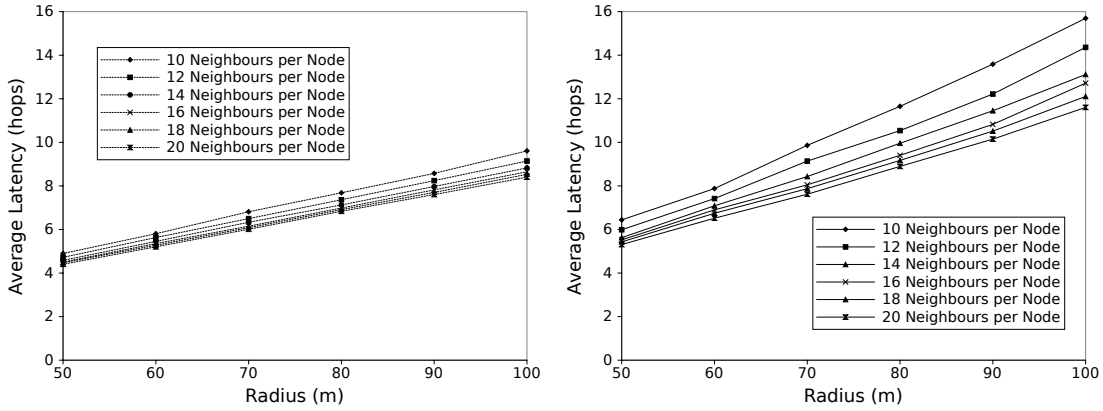
(a) MHS (b) DECOR

Figure 7.10: Removing the greedy forwarding restriction causes the max/mean ratio to increase under DECOR by a small amount but it is still significantly lower than MHS.

shows clearly that the updated version of DECOR has significantly higher latency than MHS. In fact DECOR now shows an increase of between 20.52% (±2.60%) and 63.31% (±8.06%) over MHS. It is obvious that average latency grows with radius but the results show that latency grows faster under DECOR than MHS and therefore that the increase in latency under DECOR over MHS increases (linearly) with radius ($r = 0.998$, $p = 4.71 \times 10^{-6}$). On the other hand the difference in latency is smaller at higher densities ($r = -0.973$, $p = 0.00105$).

Fig. 7.12 shows one subtree after this version of DECOR has finished. It shows that by removing the greedy forwarding limitation the subtree grows outwards to the edge of the network before turning back towards the sink again. However, it also shows that the tree becomes "twisted" with links sometimes moving in the direction of the sink and sometimes away from it. This means that many nodes in the same subtree are in range of each other but are nevertheless at different levels of the tree. This observation suggests a technique that can reduce the latency without affecting connectivity or balance.

Since balance is determined by the number of descendants of each level one node, once the routing tree is constructed the child-parent relationships can be changed without affecting balance so long as no node switches from being the descendant of one level one node to another. This makes it possible to remove the "twists" by allowing nodes to switch from their original parents to one that is closer to



(a) MHS (b) DECOR

Figure 7.11: Removing the greedy forwarding limitation results in significantly higher latency (up to 63.31% higher) compared to MHS.

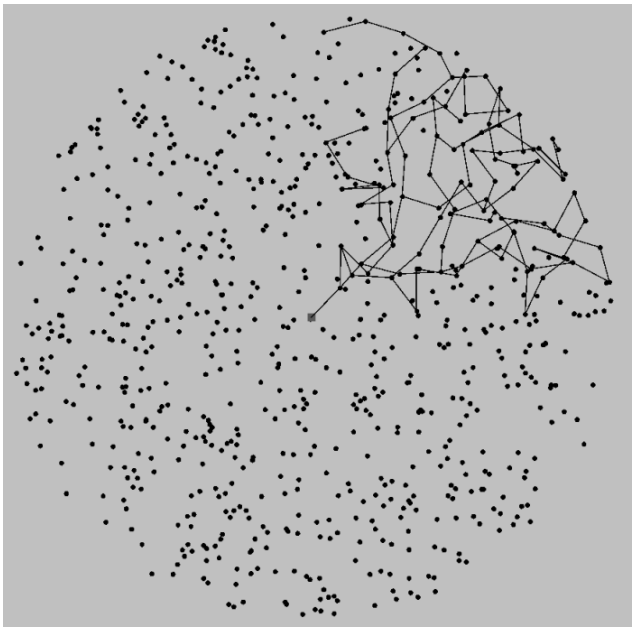


Figure 7.12: Removing the greedy forwarding requirement from DECOR results in subtrees with many "twists" and nodes may be in range of many other nodes all within the same subtree.

the sink so long as the new parent is in the same subtree as the old one.

To achieve these switches, the DECOR algorithm is modified again so that the sink creates a unique subtree ID for each of its children and these IDs are then included in the **ADV** packets so that every descendant of a given level one node has the same subtree ID. Once the DECOR algorithm is finished, a second phase is started during which the routing tree is effectively recreated. However, during this phase nodes no longer have a quota of children they can adopt and instead the only requirement is that nodes remain in the same subtree as they were in at the end of the first phase. To ensure that this happens, nodes retain their subtree ID at the end of the first phase and ignore all **ADV** packets from parents that have a different subtree ID.

An additional utility from this second phase is that the small loss of connectivity still remaining can be removed by allowing nodes that were unable to connect during the first phase (and consequently have no subtree ID) to also respond to **ADV** packets and connect to the routing tree during this second phase. This will have a small effect on balance but because the proportion of nodes that were unable to connect was relatively low the effect should be small and balance should remain high.

Fig. 7.13 shows the same subtree as Fig. 7.12 after the second phase runs. The "twists" have been replaced with "shortcuts" and a more tree-like structure is evident. Simulation results confirm that after the second phase connectivity has increased to 100% for all configurations and that balance is unaffected as seen in Fig. 7.14. However, the only reason balance is unchanged is because the changes are very small and the balance metric is not sensitive to them. The max/mean ratio, shown in Fig. 7.15, reveals that there have been some changes to balance but that the second phase has actually reduced the max/mean ratio when compared to the original DECOR algorithm. The change is small, only 9.48% ($\pm$1.49%) at its largest, and decreases logarithmically with density ($r = -0.932$, $p = 0.0068$) but increases with radius ($r = 0.99$, $p = 0.00016$). In the best case, the corresponding lifetime increase after the second phase is 85.66% compared to MHS.

The effect on latency is shown in Fig. 7.16 which, when compared to Fig. 7.11, reveals that the latency after the second phase of DECOR is closer to MHS than after the first phase. DECOR now shows between 4.99% ($\pm$1.01%) and 9.21% ($\pm$1.57%) extra latency than MHS compared to the 20.52% increase which was

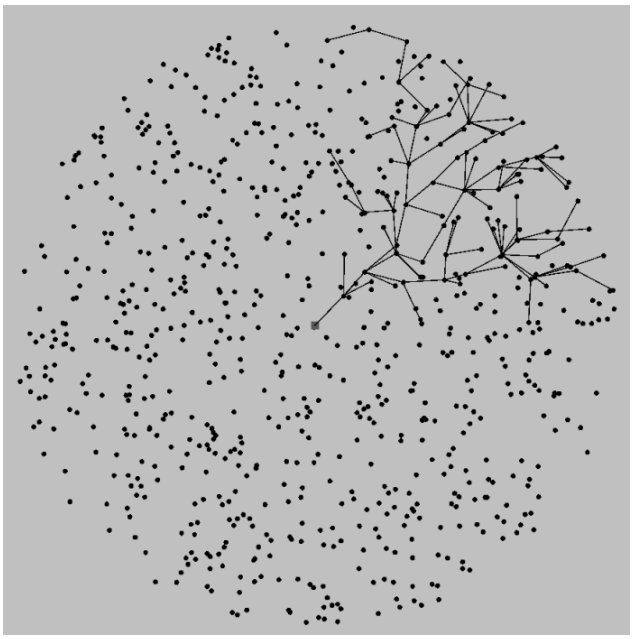
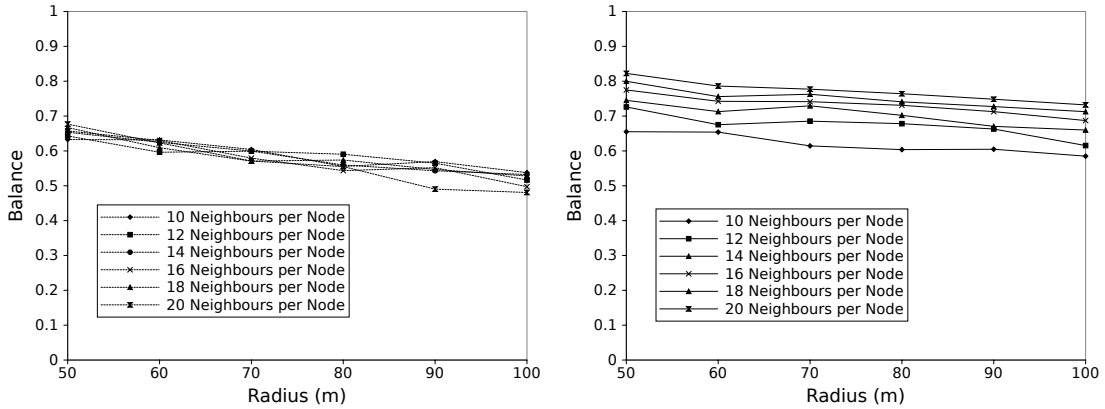
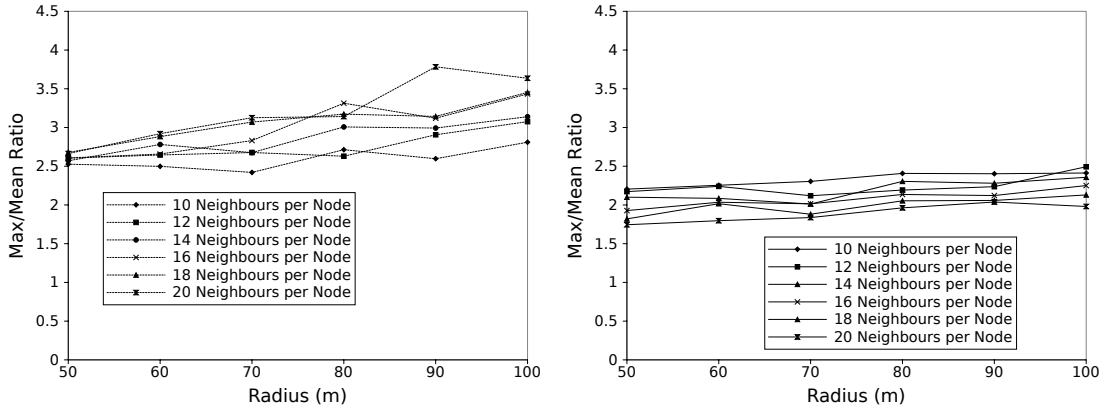


Figure 7.13: The second phase added to the end of DECOR allows the "twists" to be removed and a more tree-like structure to emerge.



(a) MHS (b) DECOR

Figure 7.14: The balance achieved by DECOR is unaffected by the second phase and remains significantly higher than MHS.



(a) MHS (b) DECOR

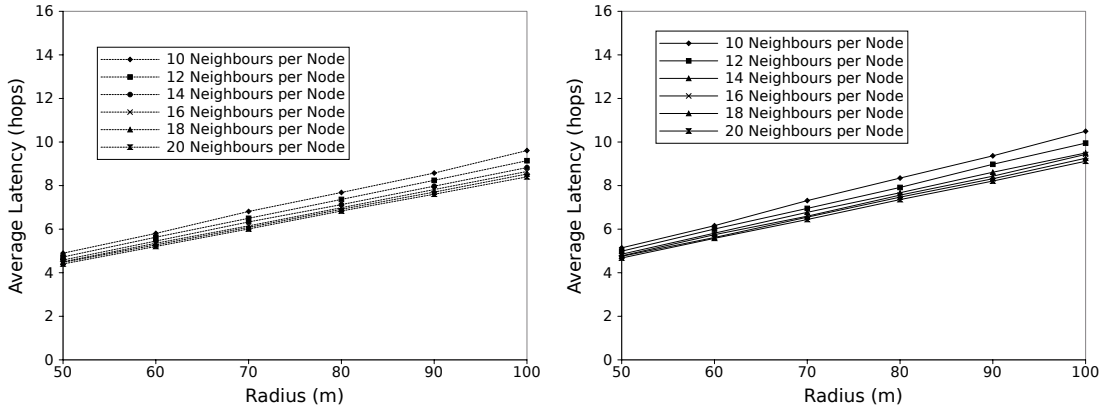
Figure 7.15: The max/mean ratio, which is far more sensitive than balance, shows a very small increase under DECOR because of the second phase but remains significantly lower than under MHS.

the smallest after the first phase. The amount of extra latency under DECOR increases with radius ($r = 0.984$, $p = 0.00036$) but is invariant with density ($p = 0.977$).

7.4 Control Overhead

The results so far have shown that the routing tree generated by the DECOR algorithm is significantly more balanced than the one produced by MHS which would result in greater lifetime. The cost is that the nodes in the tree are on average up to 10% further from the sink than in the tree produced by MHS. In this section another cost is examined, namely the number of control packets required to generate the tree. These costs are likely to be negligible compared to the network's energy usage because they are a one-off initial setup cost; nevertheless it is still useful to take them into consideration.

Fig. 7.17 shows the average number of control packets transmitted by each node. Under the MHS algorithm each node broadcasts an initial advert and then broadcasts an adoption confirmation packet for each child it adopts plus it must also transmit a packet requesting adoption. Although it appears that the average number of transmitted packets under MHS remains constant, it does in fact increase logarithmically with radius ($r = 0.976$, $p = 0.00083$) and density ($r = 0.939$,



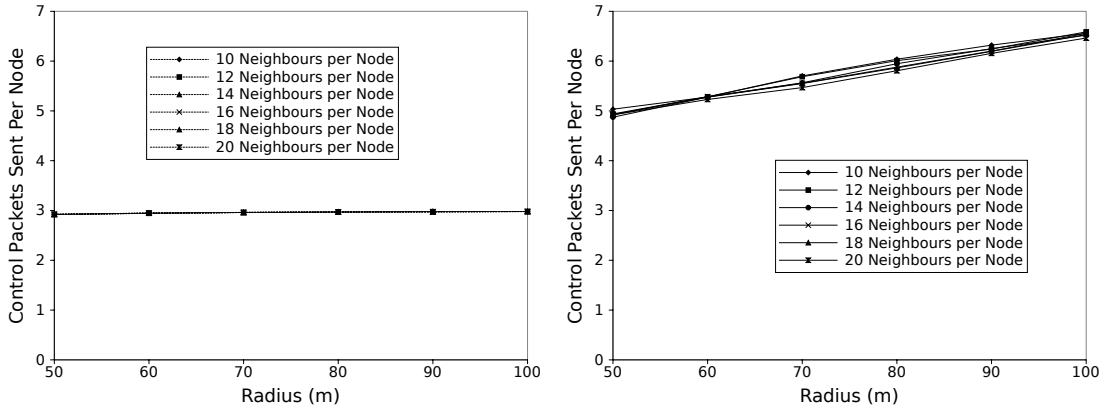
(a) MHS (b) DECOR

Figure 7.16: After the second phase of DECOR the amount of extra latency is greatly reduced and is at most 9.21% though it grows with radius.

$p = 0.0055$), approaching three.

The variation in the number of packets transmitted under MHS cannot derive from imbalance because, regardless of the balance, each node can have only one parent and therefore precisely one confirmation packet is broadcast per node regardless of the number of children that parent already has. Instead, the small variation must derive from a small number of nodes being unable to connect to the routing tree at all because of voids in the network space. The increase in the number of broadcast packets with density is the result of fewer voids and therefore fewer nodes that bring down the average. The increase with radius, on the other hand, is probably because the number of nodes caught in voids becomes an even smaller proportion of the total as the number of nodes in the network increases. This also explains why the correlation is logarithmic because the maximum number of transmissions on average under MHS is three.

In contrast to MHS, under DECOR the number of packets transmitted per node falls slightly with density ($r = -0.968$, $p = 0.00155$) but increases with radius ($r = 0.999$, $p = 3.48 \times 10^{-8}$). The fall with density is probably because the child nodes are more likely to be adopted by their first choice parent which reduces the number of packets that need to be transmitted. On the other hand, previous results already showed that the balance of the network falls as the radius increases because more nodes are unable to fill their quotas. The knock-on effect of that is that the child nodes find it harder to get adopted and must transmit more



(a) MHS (b) DECOR

Figure 7.17: The number of packets sent by each node is higher under DECOR and increases with radius whereas under MHS a node never sends more than three packets.

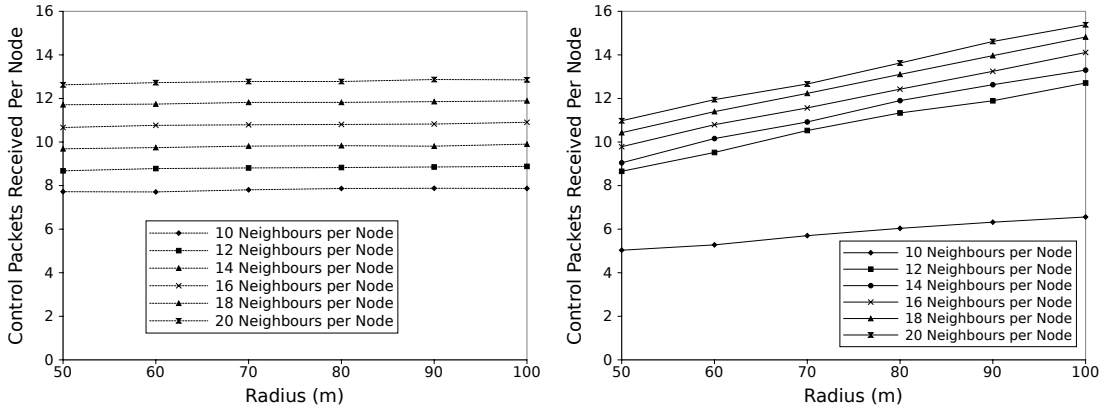
requests.

Despite DECOR requiring more packets per node, the cost is very small, no more than four extra packets per node in the worst case. Compared to the number of data packets that will be transmitted by each node over the course of the network's lifetime these extra packets are negligible.

The results for the average number of control packets received by each node are very different, as shown in Fig. 7.18. Again the number under MHS increases logarithmically, but barely, with radius ($r = 0.992$, $p = 0.00011$) but clearly increases with density ($r = 0.999$, $p = 2.21 \times 10^{-10}$). With DECOR there is an increase in the number of packets received with both radius ($r = 0.999$, $p = 5.87 \times 10^{-7}$) and density.

It is not obvious whether the results for ten neighbours per node are erroneous or whether the relationship is logarithmic. Ignoring that data point shows a very strong linear correlation ($r = 0.999$, $p = 2.31 \times 10^{-8}$) whereas the correlation, assuming a logarithmic relationship, is weaker but still significant ($r = 0.905$, $p = 0.013$).

As a result of the invariance with radius of received packets under MHS and the increase under DECOR, the difference between the two grows as radius increases. However, at the lowest radius value, the nodes using DECOR actually receive up



(a) MHS (b) DECOR

Figure 7.18: The average number of control packets received per node increases with density and, in the case of DECOR, with radius as well. However, because DECOR can aggregate many adoption confirmations into a single packet and MHS cannot, the difference in the number of packets received is not as great as the difference in the number transmitted.

to 34.78% (±4.55%) fewer packets on average than with MHS, though at the largest radius the DECOR algorithms results in up to 43.11% (±1.66%)more packets per node.

Interestingly, despite nodes transmitting more than double the number of control packets each under DECOR when the radius is 100m, the number they receive is less than 1.5 times as many. This is because under DECOR many adoption confirmations can be aggregated into a single packet whereas under MHS nodes select parents sequentially so that every adoption has its own confirmation packet which is received by all neighbours.

7.5 Chapter Summary and Conclusions

This chapter builds on the previous chapter by showing that the distributed construction of a static routing tree which aims at maximising inner-corona balance is an effective method for maximising network lifetime. This approach to lifetime maximisation is new and the results in this chapter show that, compared to the next best distributed protocol MHS, significant improvements can be made.

The DECOR algorithm works by imposing a limit on the number of children

than nodes may adopt during the tree construction phase. The limits are carefully chosen based on the predicted number of nodes in each level of the routing tree. A second phase was included to allow for complete connectivity and greatly reduce the added latency.

The simulation results in this chapter confirm that DECOR can achieve full connectivity while providing significant increases to balance which correspond to a lifetime increase of up to 85% compared to MHS. The major trade-off to achieve this is latency which increases by between 5% and 10% which is a small price to pay for such a large improvement in network lifetime.

To the best of my knowledge the two protocols proposed in the last two chapters are the only fully distributed protocols that aim to maximise inner-corona balance. The DECOR protocol proposed in this chapter has been shown to provide a very large increase in network lifetime with only a small trade-off. However, the simulated network conditions in this chapter are somewhat idealised. In the next chapter the DECOR algorithm is tested in scenarios that move beyond the simplified corona model to show that even under those conditions it still outperforms the next best protocol, MHS.

Much of the work presented in this chapter was published in [KF12c].

Chapter 8

DECOR Beyond the Corona Model

In the previous chapter the DECOR algorithm was proposed and analysed and the results showed that it could provide significant improvements to inner-corona balance for a small latency trade-off. All the preceding analysis has been based on the corona model which offers a mathematically convenient model for a sensor network but makes some simplifications and imposes constraints in order to do so. In this chapter, the DECOR algorithm is analysed in scenarios that are somewhat different from the simple corona model.

First, the unit disk graph (UDG) model is dropped and a more accurate packet reception rate model is used which is then used throughout the rest of this chapter. Second, in Section 8.2, the effect of moving the sink away from the centre of the network is considered. Although it is obvious that this will increase the latency of the network, the question is what effect it will have on DECOR's ability to produce a balanced tree. Finally, in Section 8.3, the method of deployment is changed to a Gaussian distribution and the DECOR algorithm is modified to account for a different distribution type.

8.1 Packet Reception Rate

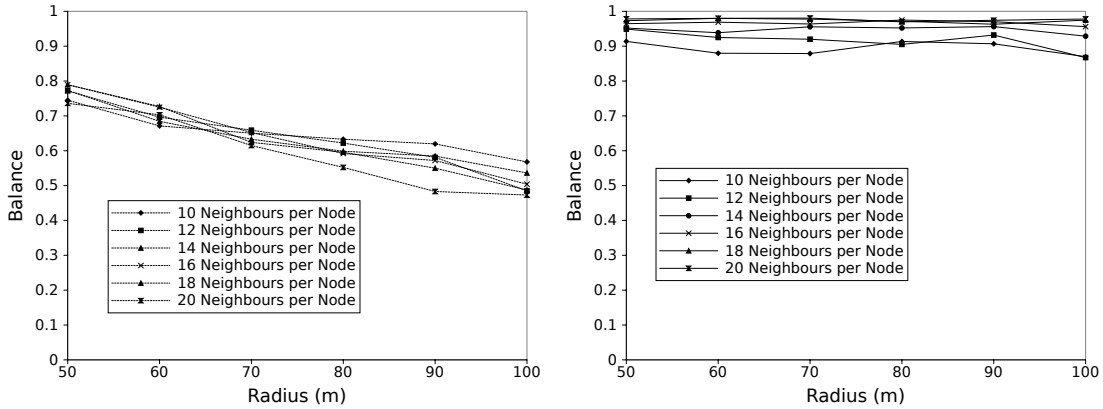
The corona model, with its fixed width coronas, is based on the UDG model which states that every node has the same, fixed transmission range. The UDG model can be thought of as predicting the packet reception rate (PRR) and predicting it to be 100% if the distance between transmitted and receiver is below some threshold and 0% if the distance is greater than that value. Although this simplification was strongly justified in Section 3.3, it remains an inaccurate model of the PRR.

In this section the more accurate packet reception rate (PRR) model derived by Zuniga and Krishnamachari is used [ZK04]. The model, shown in equation (8.1), relates the PRR to distance based on the log-normal shadowing model (see Chapter 3 for more details and Table 3.1 for the values of the variables). Following the results in that chapter, the absolute reception-based blacklisting (ARB) strategy is used by the nodes to determine the optimal relay nodes. The results using this more accurate model are in line with those found using the unit disk graph model.

$$PRR(d) = \left(1 - \frac{1}{2}\exp^{-\frac{\gamma(d)}{2}\frac{1}{0.64}}\right)^{b} \tag{8.1}$$

Fig. 8.1 shows the balance achieved by DECOR with ARB compared to MHS. It is clear that the balance achieved by DECOR is very high, ranging between 0.87 ($\pm$0.03) and 0.98 ($\pm$0.007) and is significantly higher than under MHS. An interesting result is that while the balance under MHS falls with radius, the balance achieved by DECOR shows no statistically significant variation with radius ($p = 0.135$). However, balance does increase logarithmically with density ($r = 0.978$, $p = 0.00069$). The improvement of DECOR over MHS increases with both radius ($r = 0.999$, $p = 1.63 \times 10^{-6}$) and density ($r = 0.983$, $p = 0.00042$), ranging between 22.19% ($\pm$3.96%)and 106.77% ($\pm$20.51%).

These results are reflected in the max/mean ratio shown in Fig. 8.2 which again demonstrates that the results for DECOR are significantly better than for MHS, ranging between 68.30% ($\pm$16.04%) and 300.39% ($\pm$47.06%) lower. Similarly to balance, the improvement of DECOR increases with both radius ($r = 0.991$,



(a) MHS (b) DECOR

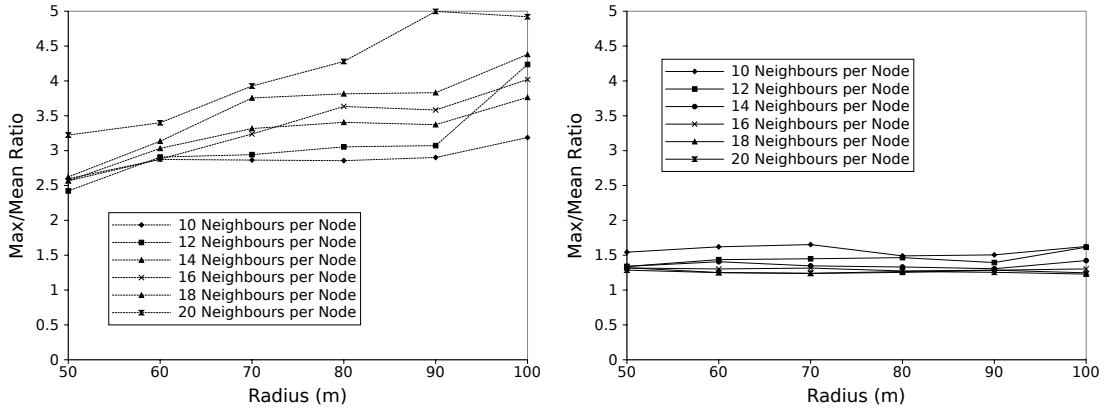
Figure 8.1: Balance is remarkably high under DECOR with a realistic PRR model, increasing logarithmically with density but not falling with radius. Overall, DECOR provides between 20% and 100% more balance than MHS.

$p = 0.00013$) and density ($r = 0.991$, $p = 0.00013$). Overall, the corresponding lifetime increase is up to 250%.

The reason for the significantly higher balance of DECOR with this model compared to the earlier results is that under ARB the average link is longer which raises the effective density of the network. All the results have shown that balance increases with density under the DECOR algorithm and therefore it is not surprising that the DECOR algorithm performs better under ARB than under UDG.

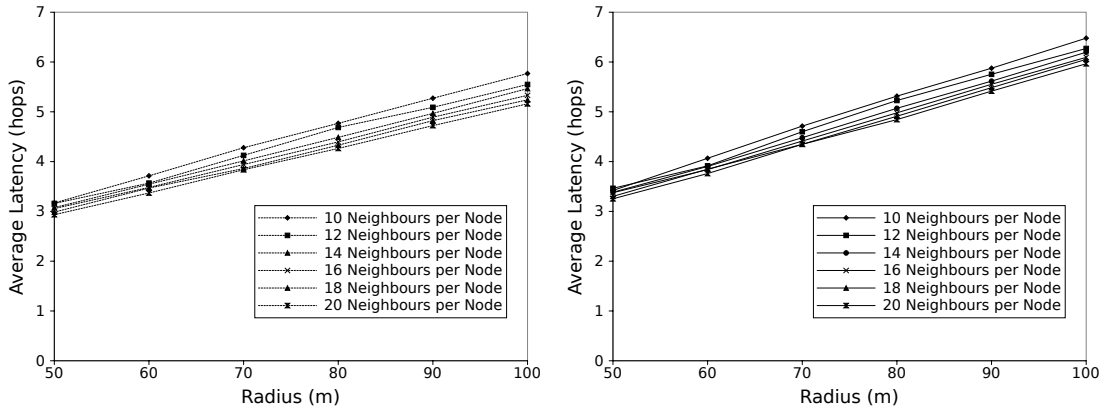
The connectivity in all configurations is 100% under both DECOR and MHS and the trade-off for latency remains, as with the unit disk graph model. Fig. 8.3 shows that the latency under DECOR is higher than under MHS. As with the earlier results, the extra latency is relatively small, between 7.35% ($\pm$0.91%) and 13.49% ($\pm$0.76%), but increases slowly with radius ($r = 0.987$, $p = 0.00025$) and density ($r = 0.988$, $p = 0.00019$).

The results for the average number of control packets sent per node are shown in Fig. 8.4. As with the results using the UDG model in the previous chapter, the number of packets sent under MHS increases logarithmically with radius ($r = 0.984$, $p = 0.000396$) but is invariant with density ($p = 0.923$). With DECOR, the number increases with both radius ($r = 0.992$, $p = 8.72 \times 10^{-5}$) and density ($r = 0.932$, $p = 0.0067$).



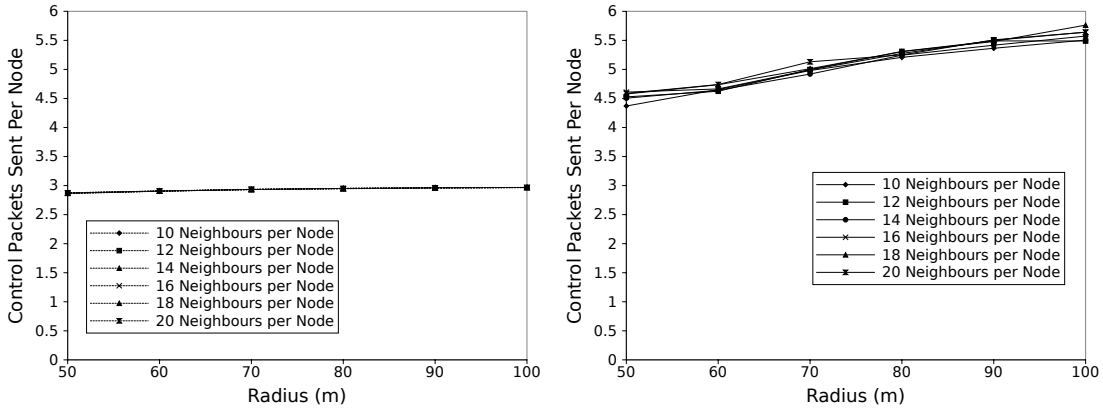
(a) MHS (b) DECOR

Figure 8.2: The max/mean ratio under DECOR is significantly lower than under MHS, showing an increased lifetime of up to 250%.



(a) MHS (b) DECOR

Figure 8.3: The trade-off for the improved balance is extra latency but these results accord with the earlier ones in showing a small increase, this time up to 13.49%.



(a) MHS (b) DECOR

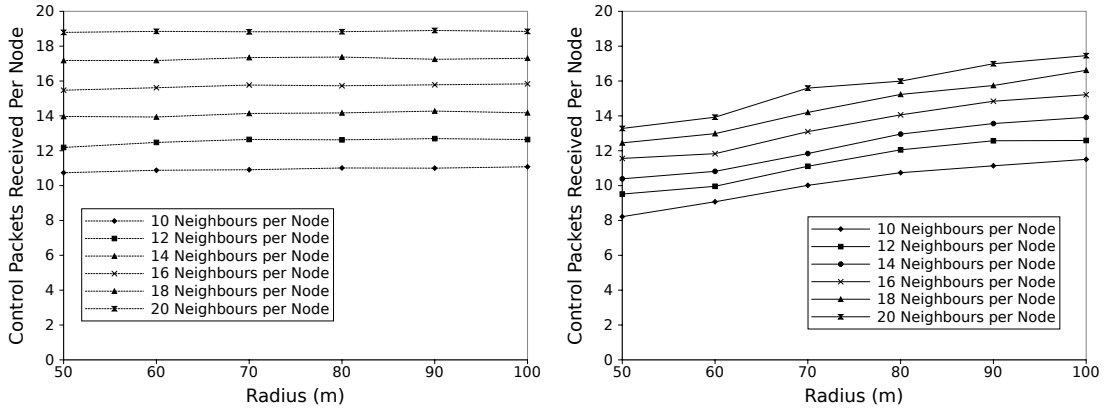
Figure 8.4: The average number of packets transmitted per node is almost constant under MHS but increases with radius under DECOR.

Fig. 8.5 shows the average number of control packets received per node. The results show a small but statistically significant logarithmic increase for MHS with radius ($r = 0.943$, $p = 0.0047$) and a larger increase with density ($r = 0.999$, $p = 8.47 \times 10^{-11}$). With DECOR the correlations with radius ($r = 0.989$, $p = 0.00019$) and density ($r = 0.999$, $p = 1.99 \times 10^{-7}$) are similarly clear. However, with the more accurate PRR model the number of packets being received per node is actually lower under DECOR than MHS. This is because the effectively higher density means that under DECOR nodes find it easier to find a parent to adopt them which reduces the number of requests they must send and hence receive, whereas under MHS the extra density means that more adoption confirmations must be overheard.

These results, with the more accurate PRR model, are in line with the earlier results and confirm that DECOR can significantly improve network balance and lifetime for a modest increase in latency.

8.2 Away From the Centre

In the corona model the sink is assumed to be in the centre of the network area which is optimal in terms of latency and balance. In this section two alternative positions are considered - edge and side - as illustrated in Fig. 8.6. The side position is likely to be worse in terms of balance than an edge positioned sink



(a) MHS (b) DECOR

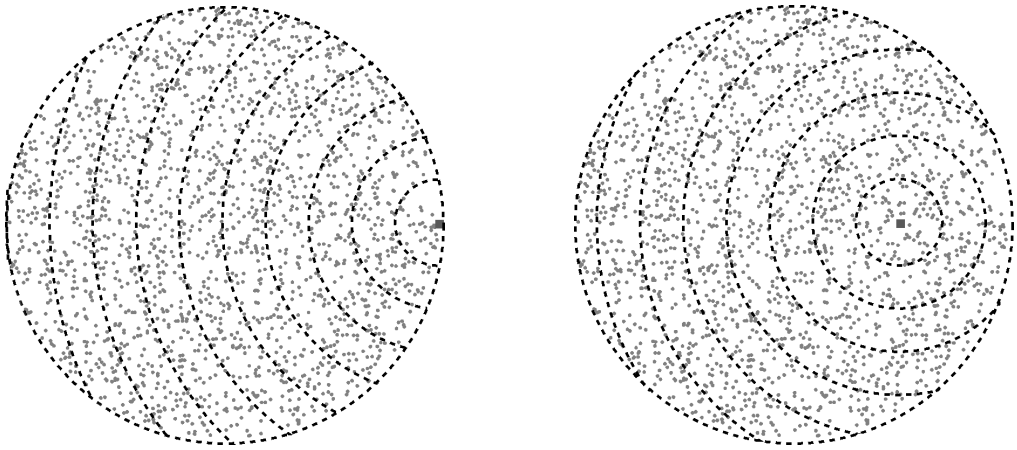
Figure 8.5: The number of control packets received per node increases with density and radius under DECOR but, for these values, is still lower than under MHS.

because the sink is placed with one quarter of the network diameter on one side and three quarters on the other. This means that there will be more nodes on one side of the sink than on the other. If the subtrees grow outward from the sink towards the network edge then intuitively the subtrees to the side of the sink with more space will be larger than those on the other side and this results in imbalance. Since the side position is liable to perform worse the edge position is analysed first.

8.2.1 Edge-Positioned Sink

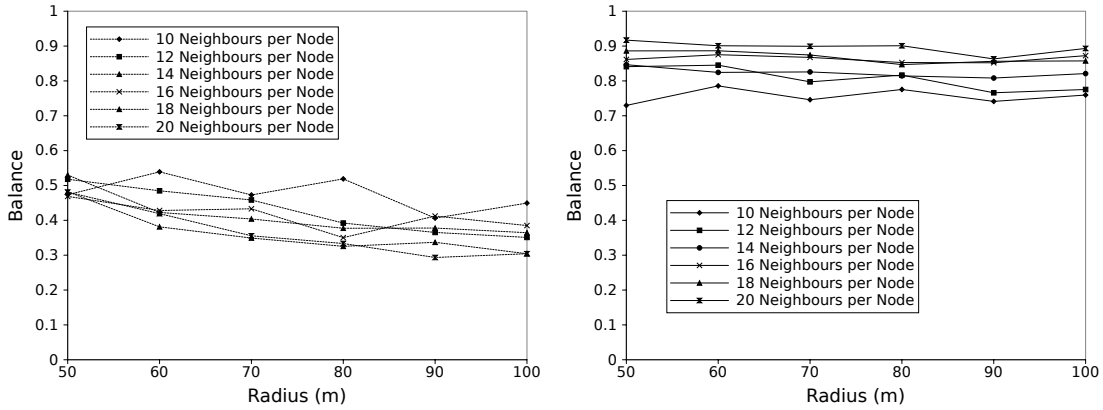
The balance with an edge positioned sink is shown in Fig. 8.7 and DECOR clearly outperforms MHS. As with the results in the previous section, there is no statistically significant change in the balance with radius ($p = 0.057$) but the balance does increase with density ($r = 0.976$, $p = 0.00086$). However, compared to the central sink, the range of balance values is lower with the highest balance falling from 0.98 to 0.917 (± 0.025). The effect on MHS is greater and the improvement of DECOR over MHS increases as a result, rising to between 45.73% ($\pm 18.44\%$) and 193.58% ($\pm 51.95\%$). As expected, the improvement increases with both radius ($r = 0.972$, $p = 0.0012$) and density ($r = 0.98$, $p = 0.00053$).

Fig. 8.8 shows the results for the max/mean ratio. As with balance, the overall



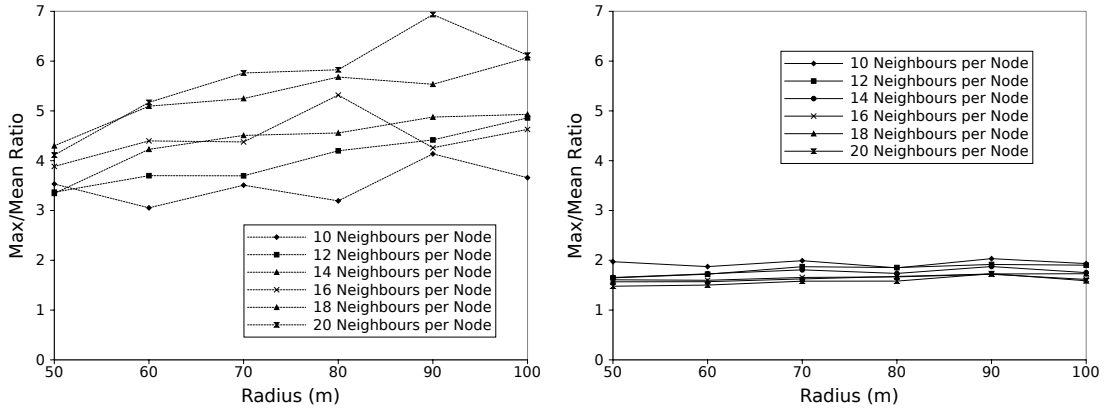
(a) Edge (b) Side

Figure 8.6: The two alternative sink positions.



(a) MHS (b) DECOR

Figure 8.7: The balance with an edge based sink is lower than with a central one but the relationship with radius and density is similar.



(a) MHS (b) DECOR

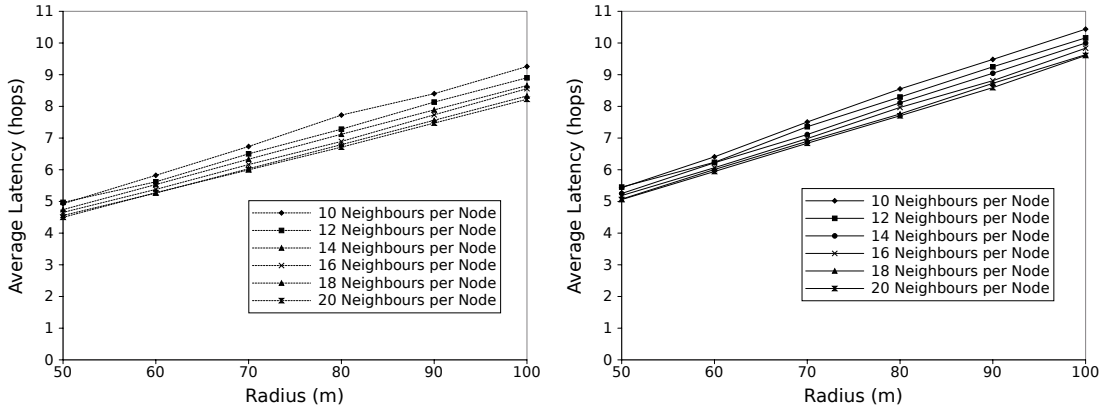
Figure 8.8: Although the max/mean ratio is higher with an edge based sink, the improvement of DECOR over MHS remains almost unchanged.

performance is worse than with a central sink. The improvement of DECOR over MHS is similar, though, to that with a central sink ranging between 63.02% (±17.09%) and 300.59% (±52.01%).

The trade-off for the gained balance is an increase in latency and this is also found with an edge based sink, as shown in Fig. 8.9. The absolute latency is obviously larger with an edge sink than with a central sink but the relative difference between DECOR and MHS is also increased slightly, now ranging from 9.71% (±1.14%) to 16.81% (±1.28%).

The results for the number of control packets sent and received are shown in Fig. 8.10 and Fig. 8.11 respectively. As with the central sink the number sent per node under MHS is nearly constant, increasing slowly and logarithmically with radius ($r = 0.981$, $p = 0.00057$) though it is invariant with density ($p = 0.29$). Under DECOR the number sent per node also increases with radius ($r = 0.994$, $p = 4.86 \times 10^{-5}$) and shows no correlation with density ($p = 0.13$).

The number of packets received per node follows the same pattern as with a central sink. Under MHS there is a logarithmic increase with radius ($r = 0.968$, $p = 0.0015$) and a linear increase with density ($r = 0.999$, $p = 9.27 \times 10^{-9}$). For DECOR the correlation with radius is linear ($r = 0.992$, $p = 8.58 \times 10^{-5}$) and there is also a strong correlation with density ($r = 0.999$, $p = 2.04 \times 10^{-6}$). With the central sink DECOR nodes received fewer packets than MHS ones but this is no longer always the case with an edge based sink. At lower radius values DECOR



(a) MHS (b) DECOR

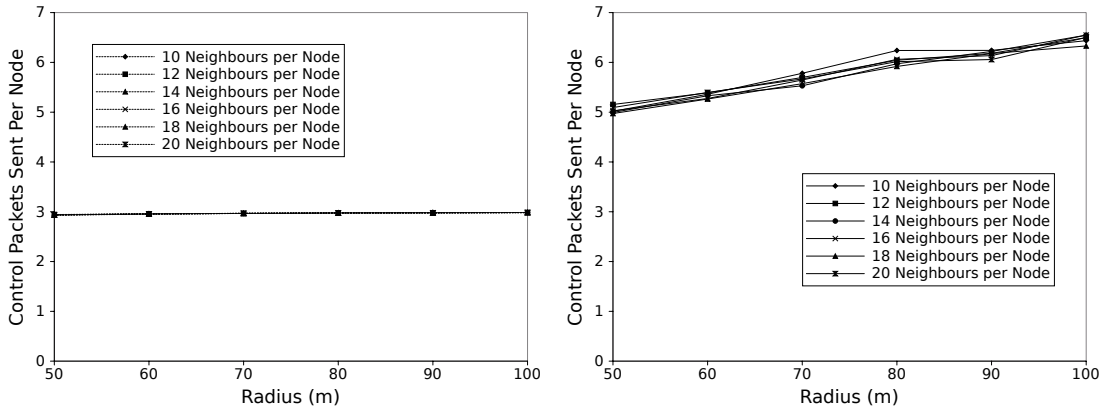
Figure 8.9: When the sink is at the edge of the network the latency is obviously increased but also the relative increase of latency with DECOR is slightly higher with a maximum value of 16.81%.

still performs better than MHS but at higher radius values MHS outperforms DECOR.

This result is unsurprising since even with the central sink it was clear that as the radius increased the relative difference in the number of received packets fell. With an edge based sink the effective radius of the network is doubled and therefore it is to be expected that MHS starts to outperform DECOR on this measure.

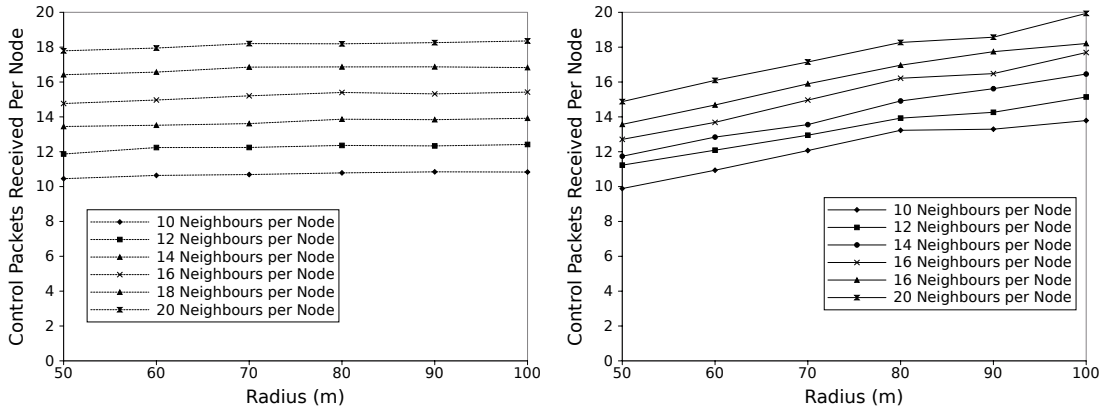
8.2.2 Side Positioned Sink

The results for the side positioned sink are very similar in pattern to those of the edge based and central sink and are as might be predicted. The balance, shown in Fig. 8.12, is lowest with the side sink because the network is less symmetrical around the sink. However, the balance is still higher under DECOR than MHS although in this case the balance actually starts to fall with increased radius ($r = -0.956$, $p = 0.0028$) which is probably because the effects of the lack of symmetry is more pronounced with a larger radius. The balance still increases with density though ($r = 0.97$, $p = 0.0013$) but the improvement of DECOR over MHS is lower than with an edge sink, ranging from 38.76% ($\pm$17.38%) to 164.59% ($\pm$36.66%). The pattern is the same as with an edge based sink, with the



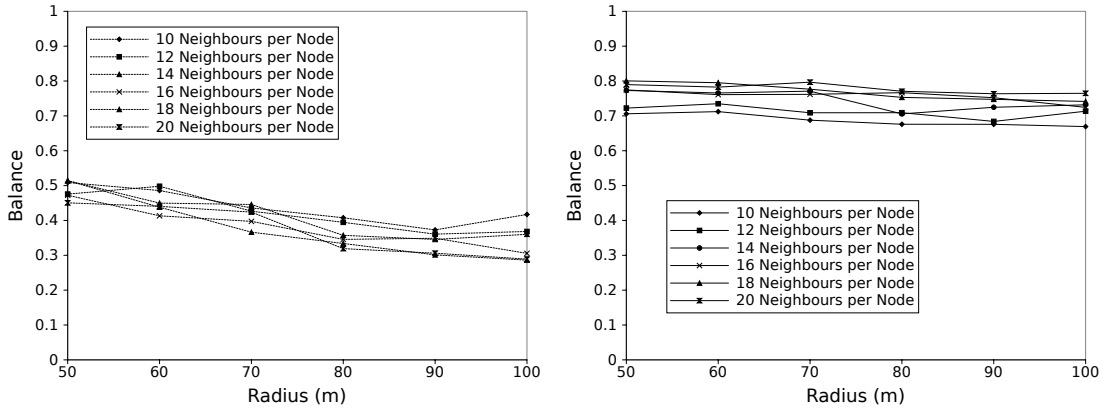
(a) MHS (b) DECOR

Figure 8.10: The number of control packets sent per node is larger under DECOR and increases with radius.



(a) MHS (b) DECOR

Figure 8.11: The number of control packets received per node under DECOR increases with radius which explains why with a central sink DECOR requires nodes to receive fewer packets per node than MHS but the opposite starts to be true with an edge based sink.



(a) MHS (b) DECOR

Figure 8.12: The balance with a side based sink is lower than with a central or edge based one but the relationship with radius and density is similar.

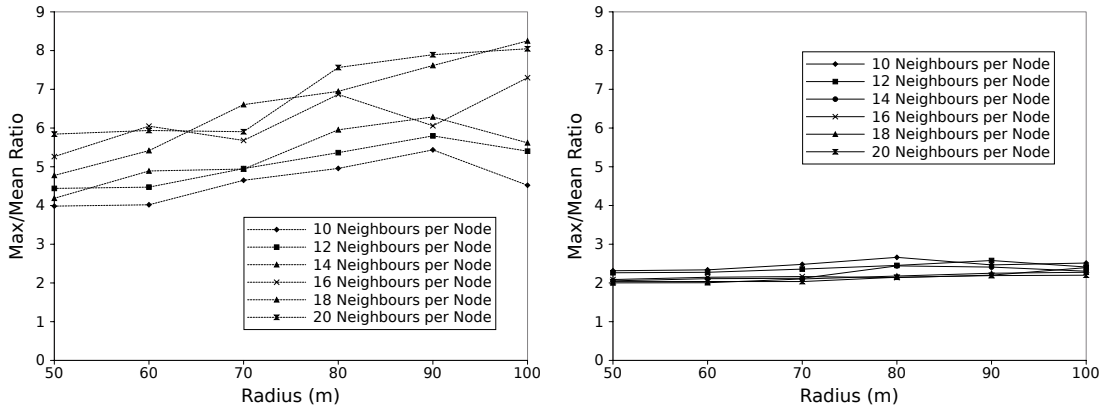
improvement increasing with both radius ($r = 0.983$, $p = 0.00043$) and density ($r = 0.985$, $p = 0.00034$).

Fig. 8.13 shows the results for the max/mean ratio. Again the results are worse than with the edge based sink but nevertheless DECOR performs better with an improvement of up to 265.78% (±50.48%).

The latency trade-off shown in Fig. 8.14 is obviously similar to with a central or edge based sink. Although the absolute latency values are lower with a side sink than with an edge sink, the relative extra latency results are nearly identical, ranging from 9.64% (±1.37%) to 16.26% (±1.14%).

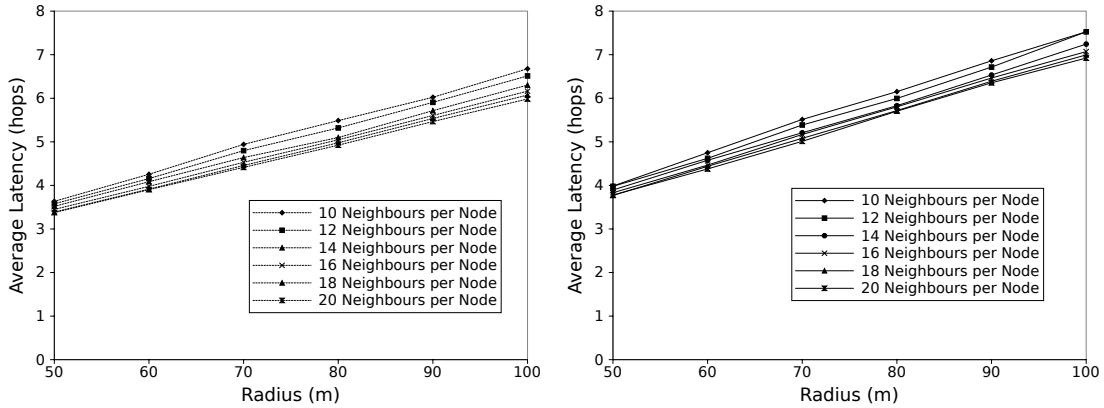
The same pattern can be seen with the number of control packets sent and received per node shown in Fig. 8.15 and Fig. 8.16 respectively. The patterns are the same but the absolute values are lower. With the number of packets received per node the results are better under DECOR than MHS but once again the improvement falls with radius ($r = -0.998$, $p = 5.12 \times 10^{-6}$) but increases with density ($r = 0.979$, $p = 0.00068$) which means that at the larger radius values and lower densities, the number received per node is slightly lower under MHS than DECOR.

The results in this section show that the DECOR algorithm continues to outperform the next best protocol, MHS, even when the sink is moved away from its optimal position at the centre of the network. When the sink is not in the centre,



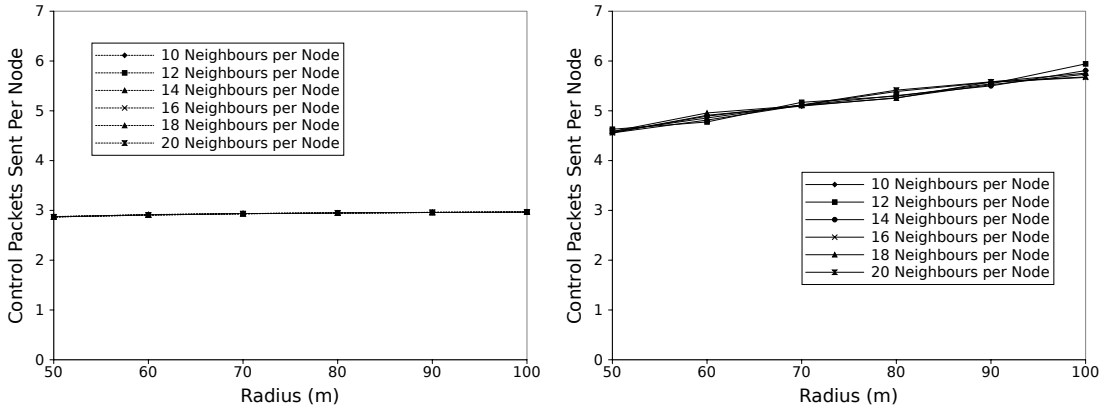
(a) MHS (b) DECOR

Figure 8.13: The max/mean ratio is lower under DECOR than MHS but the absolute values are higher for both.



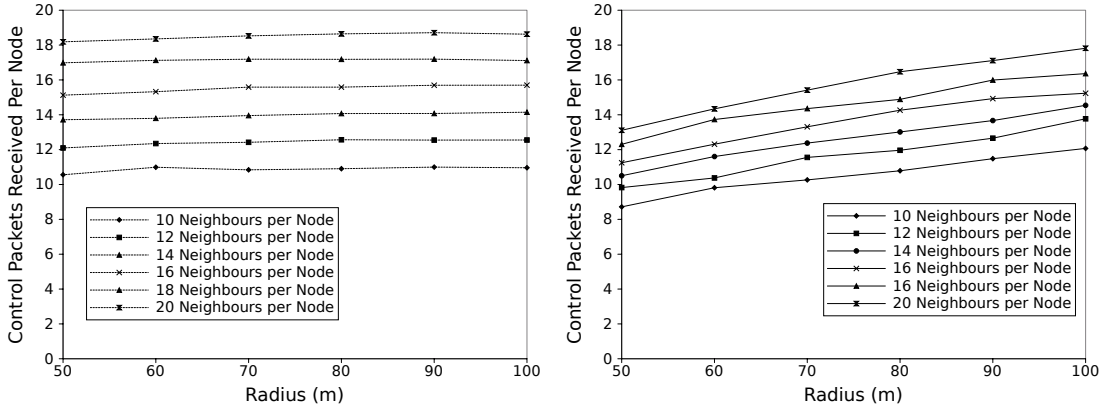
(a) MHS (b) DECOR

Figure 8.14: The latency is lower with a side sink than with an edge sink but the relative performance of DECOR and MHS are virtually identical.



(a) MHS (b) DECOR

Figure 8.15: The number of control packets sent per node is larger under DECOR and increases with radius.



(a) MHS (b) DECOR

Figure 8.16: The number of control packets received per node under DECOR increases with radius and density and so at lower radius values DECOR outperforms MHS but at higher radius and lower density values this changes.

the predicted number of nodes per level does not apply and yet DECOR still performs well with balance remaining high. In the next section the method of distribution of nodes is changed from uniform to Gaussian and DECOR is tested under those circumstances to show that its underlying principles can be adapted to different distributions.

8.3 Gaussian Distribution

The corona model and all the calculations in previous sections have all assumed that the nodes are distributed randomly and uniformly so that the density is approximately constant across the network area. In this section that assumption is replaced and the nodes are assumed to be distributed with a Gaussian distribution whose mean is the centre of the network (where the sink is) and with standard deviation equal to half the radius. Fig. 8.17 illustrates a network whose nodes are distributed according to the Gaussian distribution.

While the uniform distribution is optimal in terms of coverage and connectivity as discussed in Section 3.1.4, the Gaussian distribution mimics the kind of deployment that might be expected if the nodes are deployed from a central point. For example, if nodes are dropped from a hovering helicopter that remains static during the drop it is reasonable to suppose that there will be more nodes close to the helicopter's position than further away. For this reason the Gaussian distribution is considered in this section as an alternative to the uniform.

The DECOR algorithm revolves around the level quotas which were initially calculated based on the uniform distribution where the number of nodes per level is easily approximated. For the Gaussian distribution it is assumed that the designer knows approximately how many nodes will be in each corona before deployment which can be used to calculate the level quotas. The underlying concept is to select a quota that minimises the number of disconnected nodes in the next corona. It should be noted that a quota that predicts no disconnections at all is too high because it affords no space for errors and is likely to result in lower balance.

The algorithm for calculating the level quotas is given in algorithm 8.1. During the calculation, the ratio of the number of nodes in each level to the number in

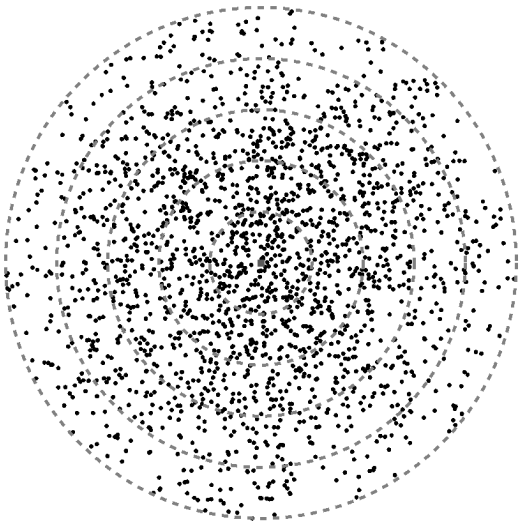


Figure 8.17: A sensor network with Gaussian distributed nodes.

the first level is used and all numbers in the pseudocode are actually ratios.

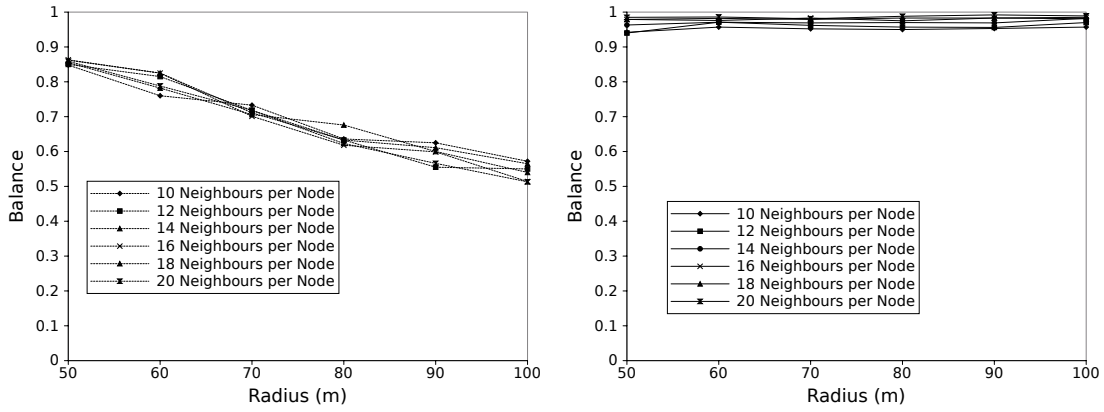
Fig. 8.18 shows the results for balance and it is clear that the network is more balanced under both MHS and DECOR. This is because the Gaussian distribution results in more nodes near the centre which makes it easier to balance the workload and reduces the impact of imbalance. The balance under DECOR ranges from 0.94 ($\pm$0.02) to 0.989 ($\pm$0.003). There is no significant correlation with radius ($p = 0.084$) but balance improves with density ($r = 0.975$, $p = 0.00092$). With MHS the story is quite different and the balance falls with radius ($r = -0.994$, $p = 5.21 \times 10^{-5}$) but there is no significant correlation with density ($p = 0.505$). The result is that DECOR provides between 10.59% ($\pm$4.51%) and 92.79% ($\pm$15.87%) more balance than MHS and this balance increases with radius ($r = 0.997$, $p = 1.04 \times 10^{-5}$) but shows no significant correlation with density ($p = 0.053$).

The max/mean ratio shown in Fig. 8.19 follows a similar pattern to previous results. The ratio is low with DECOR and decreases slowly with radius ($r = -0.955$, $p = 0.003$) and with density ($r = -0.988$, $p = 0.00022$). On the other hand, under MHS the ratio increases with radius ($r = 0.993$, $p = 7.37 \times 10^{-5}$) and density ($r = 0.981$, $p = 0.00054$). Taken together the result is that DECOR has a ratio between 39.74% ($\pm$6.13%) and 78.79% ($\pm$2.29%) lower than MHS.

The trade-off for latency, seen in Fig. 8.20, is also similar to previous results,

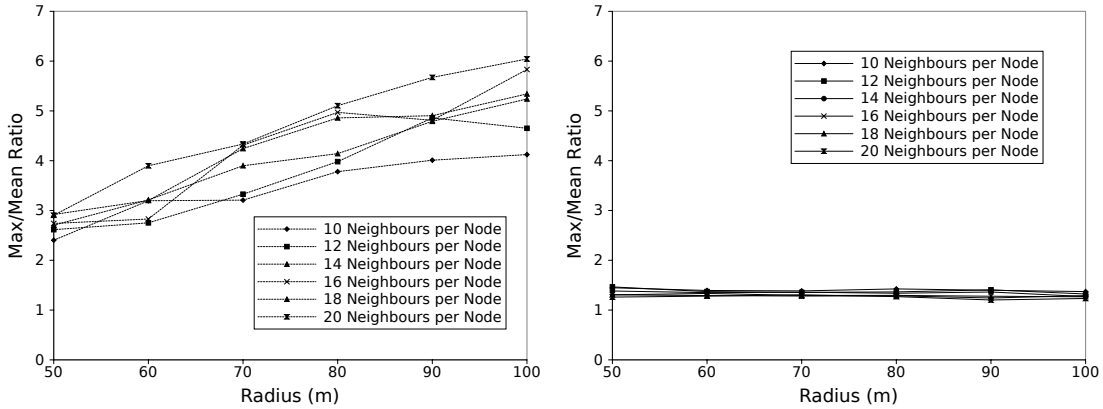
Algorithm 8.1 Level Quotas

```
1: function LEVELQUOTAS(nodes,sink,levels)
2:     connectedSoFar = 1.0
3:     for all level ∈ levels do
4:         finished = false
5:         tmpQuota = 0
6:         while !finished do
7:             tmpQuota++
8:             connected = connectedSoFar×tmpQuota
9:             disconnected = levels[level+1].ratio - connected
10:            if disconnected ≤ 0 then
11:                finished = true
12:                level.quota = tmpQuota - 1
13:                if levelquota < 1 then
14:                    levelquota = 1
15:                end if
16:                connectedSoFar = connectedSoFar×quota
17:            end if
18:        end while
19:    end for
20: end function
```



(a) MHS (b) DECOR

Figure 8.18: The DECOR algorithm can adapt itself to a Gaussian distribution and provide very high balance.



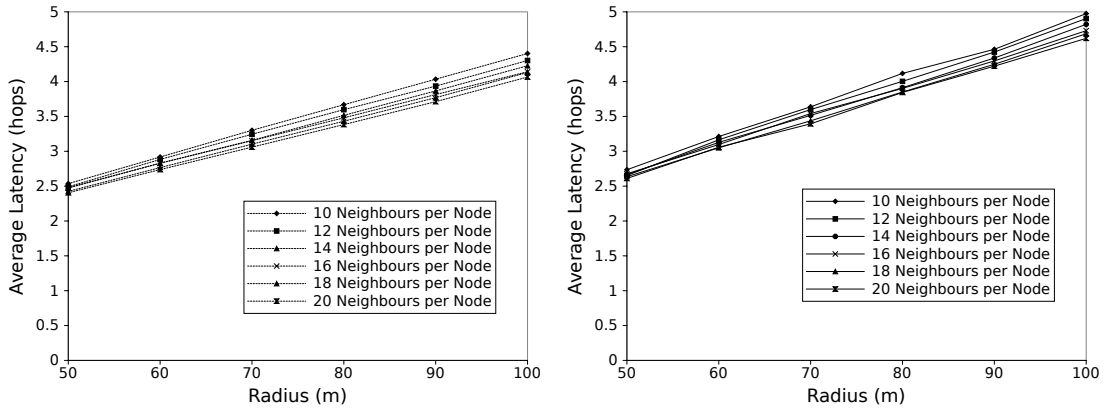
(a) MHS (b) DECOR

Figure 8.19: The max/mean ratio is much lower under DECOR than MHS leading to a reduction of up to 78.79%.

with an increase of between 6.55% (±1.28%) and 14.05% (±0.98%). It is worth noting that the latency is significantly lower with the Gaussian distribution than with the uniform one because more nodes are distributed towards the centre of the network.

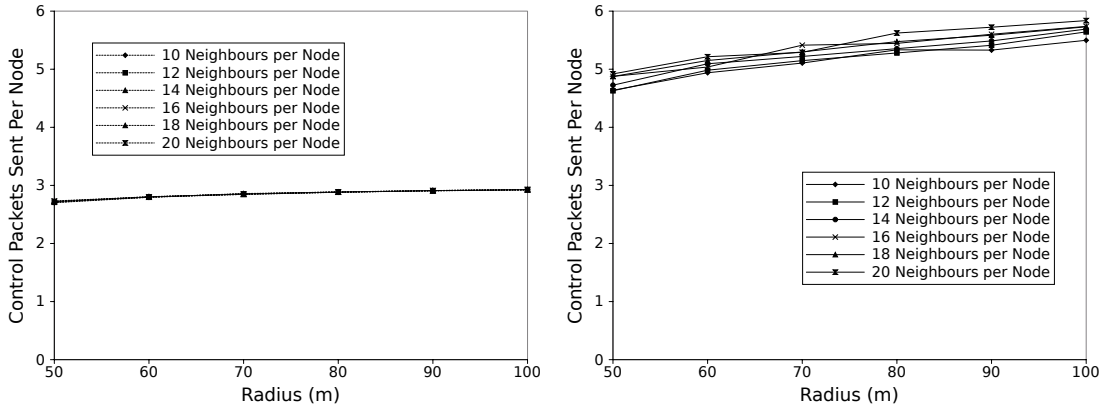
The final set of results for control packets sent and received, shown in Fig. 8.21 and Fig. 8.22 are also in line with previous results. Under MHS the number of packets sent per node increases logarithmically with radius approaching three ($r = 0.983$, $p = 0.00042$) but showing no correlation with density ($p = 0.27$). With DECOR there is an increase with both radius ($r = 0.988$, $p = 0.00023$) and density ($r = 0.983$, $p = 0.00042$).

When it comes to control packets received, MHS shows a debatably significant logarithmic correlation with radius ($r = 0.823$, $p = 0.044$) but a clear increase with density ($r = 0.999$, $p = 1.6 \times 10^{-8}$). With DECOR the increase with radius is more certain ($r = 0.988$, $p = 0.0002$) and there is also the increase with density ($r = 0.999$, $p = 5.78 \times 10^{-7}$). Because the nodes are closer to the centre in the Gaussian distribution, the effective radius is somewhat diminished which results in DECOR requiring fewer control packets to be received than MHS.



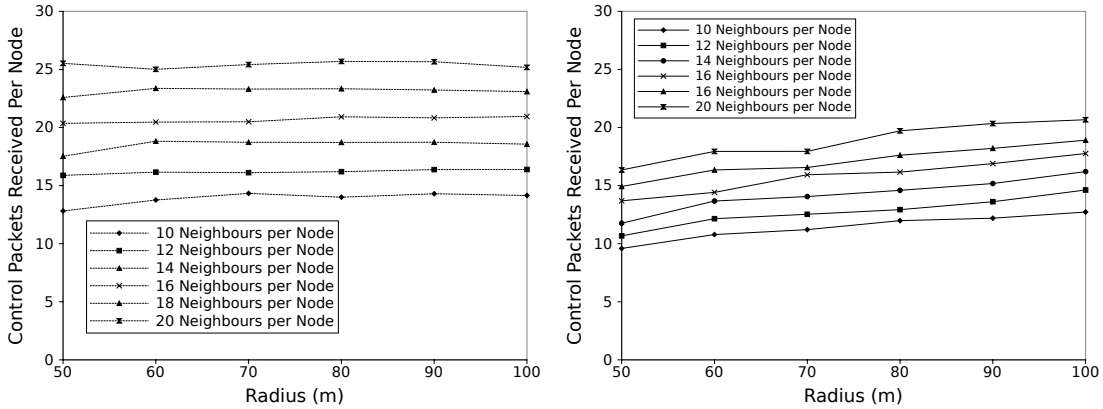
(a) MHS (b) DECOR

Figure 8.20: The increase in latency resulting from DECOR is of a similar level to that found in previous results.



(a) MHS (b) DECOR

Figure 8.21: The pattern of control packets sent per node is the same for the Gaussian distribution as for the uniform one.



(a) MHS (b) DECOR

Figure 8.22: The pattern of the control packets received per node is similar to previous results but because the effective radius of the network is smaller DECOR outperforms MHS.

8.4 Chapter Summary and Conclusions

In this chapter the DECOR algorithm has been analysed in a number of scenarios extending beyond the simple corona model. First and most importantly, the simplified unit disk graph model was replaced with a more accurate model for the packet reception rate. The sink was also moved away from the optimal position at the centre of the network to both the edge and a side position. Finally, the uniform distribution of nodes was replaced with a Gaussian one.

In all cases it was found that DECOR significantly outperformed the alternative MHS and that the trade-off with latency was stable. This proves that the underlying logic behind DECOR is not reliant on the simplifications that enabled easy analysis. In particular, in the case of the Gaussian distribution, the results showed that the logic of DECOR can be adapted to a different distribution than it was designed for. It may be possible, therefore, to adapt DECOR to many other distributions.

The results in this chapter have shown that the DECOR protocol is well suited to a range of networks and is a viable algorithm for use in real world applications. This opens up a new method for tackling the challenge of lifetime maximisation in sensor networks through the use of distributed algorithms to construct routing trees that maximise inner-corona balance.

Chapter 9

Conclusions

This thesis focused on constructing a static routing tree that would maximise the lifetime of a network suffering from the energy hole problem. A large class of sensor networks have static, homogeneous nodes which use multi-hop communication to route regularly sensed data to a single central sink. The result is a build up of traffic towards the centre of the network and an inherent imbalance in the workload in which the nodes that can communicate directly with the sink deplete their batteries much sooner than other nodes. This is the energy hole problem and is unavoidable for these networks.

The problem can be mitigated by balancing the excess work as much as possible among the most critical nodes. In this thesis this type of load balancing is referred to as inner-corona balancing and the primary aim of the thesis was to propose novel, fully distributed routing protocols that would create a static routing tree with maximised inner-corona balance. Although protocols have been proposed to maximise this type of load balancing (see Chapter 2), this is the first time fully distributed protocols have been proposed to do so.

Fully distributed protocols have been proposed that maximise a different type of load balancing which is called degree balancing in this thesis. However, this type of load balancing can never guarantee to result in perfect inner-corona balance, even in the most ideal and unrealistic scenarios. Simulation results show that they certainly improve balance and lifetime when compared to naive algorithms but without directly focusing on inner-corona balance this approach cannot be as effective as one that does (see Chapter 5).

Two protocols were proposed both based on controlling the number of children adopted by nodes during the construction of the routing tree. The first, ROBAR, did this by assigning roles to nodes and imposing rules on how new nodes attained their roles. The roles were explicitly linked to a maximum number of children that that node was allowed to adopt (see Chapter 6). The second, and more successful, protocol was DECOR which assigned quotas to nodes such that all nodes at the same level of the routing tree had the same quota and nodes could not adopt more than their quota. Techniques were added to improve on the basic version of this approach and the final result showed that very large improvements could be made to inner-corona balance, and hence network lifetime, in exchange for a small increase in the average number of hops between nodes and the sink. This approach was tested in a wide range of scenarios and its performance characteristics remained unchanged in all. It was able to adapt also to a move away from the standard uniform distribution of nodes to a Gaussian one (see Chapters 7 and 8).

There were a number of other supporting contributions made during this thesis. The question of the most energy efficient method for position-based routing was revisited in light of a key observation concerning the nature of ARQ. It was shown that the absolute reception-based blacklisting (ARB) approach was actually more energy efficient than the PRR×distance metric-based approach that had previously been considered optimal (see Chapter 3). This result was used to provide strong justification for the use of the unit disk graph (UDG) model in simulations of sensor networks as a close approximation to the performance of ARB. This is important because the UDG model is well-known to be significantly inaccurate and yet remains widely used (see Chapter 3).

The relay hole problem was also analysed for the first time in this thesis and its impact was found to be not only significant but also difficult to remove. The effect of the problem is to increase latency above optimal and reduce energy efficiency. It is also likely that the relay hole problem interferes with the effectiveness of the proposed routing protocols (see Chapter 4).

9.1 Future Work

There is a lot of scope for future work based on the contributions in this thesis and some avenues are discussed here.

The analysis surrounding ARB warrants significant expansion to consider other routing methods. In the initial analyses by Seada *et al.*, a number of different blacklisting based schemes were discussed [SZHK04]. These include relative reception-based and both absolute and relative distance based schemes. The initial analysis found that the optimal strategy was to avoid blacklisting and use a cost metric based approach and in particular to use PRR×distance as the cost metric. It therefore made sense to compare ARB to this cost metric strategy in light of the new analysis of the link cost. However, a comparison against all the alternatives would be more thorough and the conclusions drawn would be more reliable.

Furthermore, there are unanswered questions regarding the performance of ARB that were not investigated because it was previously considered to be a sub-optimal approach. These include finding the conditions required for ARB to perform to a given standard. It is well understood, for example, that a blacklisting based approach runs the risk of providing extremely poor performance if it begins to blacklist the only available routing paths. However, the ARB strategy has not been properly tested to reveal the conditions under which its performance begins to degrade.

A second area for future work is to fully understand the relationship between ARB and UDG. In this thesis a major contribution was made by showing that UDG closely approximates ARB because this justifies the continued use of UDG. However, while it was demonstrated that in principle any UDG-based simulation can be converted into a more accurate ARB-based one, the method for that conversion has not been investigated. Such a method would be extremely valuable for validating simulations and easing the interchange between mathematical analysis using the simplified UDG model and more accurate simulation using the ARB method.

The reasoning behind connecting ARB to UDG was that the average link length under ARB converges and can be used as the threshold distance in UDG. This is based on the Gaussian nature of the relationship between link length and the

packet reception rate (PRR) of the link. It therefore seems reasonable to assume that it would be possible to mathematically derive the average link length given the PRR threshold used in ARB by reversing the PRR model.

The analysis of the relay hole problem is also an area for future work. In this thesis the problem was only analysed in the context of the corona model which, although widely used, is simplified. However, the problem is not just a product of that model; it exists in reality and analysis of its properties should be conducted using more accurate PRR models. In the corona model the nodes take up no physical space but clearly in reality they do. Therefore, perhaps one method for approaching analysis of the relay hole problem with a more accurate PRR model is to compare a network to an idealised one in which every part of the network area is covered by a node of some fixed size. Another method might be to consider a node to have the relay hole problem if it cannot find a relay inside its transitional region and is forced to use only the connected region.

Solutions have been found to the problem of routing holes in sensor networks and therefore it is reasonable to hope that, if a proper analysis is conducted, solutions may be found to the relay hole problem as well. Even if the problem cannot be completely avoided there may be techniques that can reduce its impact. Doing so would reduce latency and improve energy efficiency both of which are important performance measures. A potential solution is to extend a node's reachable area by allowing it to use a relay with a lower PRR than would otherwise be acceptable. This would reduce the number of hops but since the link is less reliable, resulting in retransmissions and delays, it is not at all certain that this would serve to reduce the overall latency. Cooperative transmission between the node with the relay hole problem and a near neighbour may also serve as a basis for a solution.

The proposed ROBAR protocol was found to suffer from decreased connectivity and the method suggested for solving that resulted in reduced balance. However, the techniques applied to DECOR, namely relaxing the greedy forwarding requirement and introducing a second phase, were successful at increasing connectivity and reducing added latency. These techniques should be applied to ROBAR to determine whether they have a similar effect on that protocol. While it seems unlikely that ROBAR would be as effective as DECOR even if the techniques improved it, they may raise the performance of the ROBAR protocol above DECOR in certain situations. Assuming that the loss of connectivity under ROBAR can

be avoided without too great a cost to balance and latency, a direct comparison between ROBAR and DECOR would be warranted and beneficial.

Finally, the proposed routing protocol, DECOR, can be extended to cover other scenarios not considered in this thesis. There are many unanswered questions regarding its performance in other situations. For example, how much node mobility can it tolerate and at what point does its performance degrade below that of alternatives? Can it be adapted to situations where accurate information on node positions is not available? Does the entire routing tree need to be recreated if nodes fail or if new nodes are deployed? How would the algorithm need to be modified to handle nodes with differing initial energy levels?

9.2 Concluding Thoughts

This thesis met its primary aim through the DECOR algorithm proposed in Chapter 7. Despite the plethora of routing algorithms that have been proposed for sensor networks, DECOR is not a tweak of an existing idea. Rather it is the first (along with ROBAR proposed in Chapter 6) protocol for sensor networks that combines three important properties: it is (1) fully distributed, (2) creates a static routing tree and (3) maximises inner-corona balance.

DECOR is certainly an important contribution to the field of routing in sensor networks and is appropriate for a large number of sensor network configurations. However, clearly it is not suitable for all sensor networks and, indeed, it is certain that there is no single protocol that is optimal for all networks. Returning to the quotes at the very beginning of this thesis, the problem is that the design space for sensor networks is so large that it is virtually impossible to actually define a sensor network. How then can a single routing protocol be found that is optimal or even effective in all cases?

Nevertheless, the over-specialisation of routing protocols in this field would appear to hamper the development of sensor networks. It is reasonable to suppose that Wi-Fi would not be anywhere near as ubiquitous today were it not for the existence of a single (or small number depending on how you count) of protocols. If users were advised to select a different protocol for every different configuration of personal area networks it is hard to imagine that there would be many

such networks in existence. Yet, this is how things are in the area of sensor networks.

Perhaps one of the reasons for this is that the very simplest of routing protocols, for example restricted flooding, were not seriously tested and examined. It was noted that they were inefficient and so immediately alternatives were sought which were optimised for certain configurations. A return to first principles to discover the true extent to which simple, almost naive, protocols are capable of meeting the demands of sensor networks may well be warranted. Another approach may be to introduce highly adaptable protocols that can self-optimise to an extremely large range of networks. Techniques from machine learning or multi-agent systems may be available or extended to make this effective.

What appears undeniable is that so long as the thrust of research into routing in sensor networks is on tweaking existing ideas so as to provide optimised routing in niche networks, it will remain extremely difficult for sensor networks to become widely adopted. I believe that efforts should be redirected to evaluating protocols in a bid to find a small number that are "good enough" for as wide a range of different networks as possible. This effort should be complemented by a move towards making protocols that self-optimise so that sensor networks can finally become truly self-organising and autonomous as they were originally conceived.

Appendix A

Derivation of Equation (5.3)

In Section 5.2, it was proved that in order for degree balancing to produce perfect inner-corona balance, the $2n$ doubles that are created in every level must be drawn precisely two from each top subtree. In this appendix the probability that this happens in a given corona is derived.

This problem is identical to a ball picking problem without replacement: Suppose that an urn contains x balls, comprised of an equal number of balls of y colours, i.e. there are $z = \frac{x}{y}$ balls of each of the y colours. $2y$ balls are drawn at random without replacement. What is the probability that, of the $2y$ selected balls, there are precisely two of each colour?

To illustrate the solution, consider the case when there are three colours: red, green and blue and there are four balls of each colour. In total, there are 12 balls and six are selected. What is the probability of selecting exactly two reds, two greens and two blues.

The solution is to consider the probability of a given, ordered, combination of the desired balls, e.g. RGBRGB. This is:

$$P(O) = \left(\frac{4}{12}\right) * \left(\frac{4}{11}\right) * \left(\frac{4}{10}\right) * \left(\frac{3}{9}\right) * \left(\frac{3}{8}\right) * \left(\frac{3}{7}\right) \tag{A.1}$$

However, the order of selection is of no concern, so therefore the result must be multiplied by the number of distinct combinations of the six selected balls. In total there are 6! combinations but because there are two balls of each colour, some

of these combinations are not distinct. Therefore, the total number of combinations is divided by the number of combinations of the repeated elements, which is 2! for each colour. The total number of distinct combinations is $\frac{6!}{(2!)^3}$.

To generalise, if there are a total of x balls, made up of z balls of y different colours and $2y$ balls are selected at random, the probability that the selection contains precisely two balls of each colour is:

$$P = \frac{z^y(z-1)^y}{x!/(x-2y)!} \times \frac{(2y)!}{(2!)^y} \tag{A.2}$$

The translation to the load balancing scenario is straightforward. Each node is the equivalent of a ball. The n subtrees are the y colours. In each level of the routing tree there are $2i-1$ nodes, the equivalent of x. Finally, each subtree has $2i-1$ nodes in level l_i, the equivalent of z. Therefore, the probability that the doubles in level l_i are perfectly assigned to subtrees is the probability of perfect balance in that level, β_i:

$$P(\beta_i) = \frac{(2i-1)^n(2i-2)^n(2in-3n)!}{(2in-n)!}\frac{(2n)!}{(2!)^n} \tag{A.3}$$

Bibliography

[APZY+09] P. Andreou, A. Pamboris, D. Zeinalipour-Yazti, P.K. Chrysanthis, and G. Samaras. Etc: Energy-driven tree construction in wireless sensor networks. In *Mobile Data Management: Systems, Services and Middleware, 2009. MDM'09. Tenth International Conference on*, pages 513–518. IEEE, 2009.

[AVE08] T. Ahonen, R. Virrankoski, and M. Elmusrati. Greenhouse monitoring with wireless sensor network. In *Mechtronic and Embedded Systems and Applications, 2008. MESA 2008. IEEE/ASME International Conference on*, pages 403 –408, oct. 2008.

[BAS05] C. Buragohain, D. Agrawal, and S. Suri. Power aware routing for sensor databases. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1747–1757. IEEE, 2005.

[BBB09] F. Bouabdallah, N. Bouabdallah, and R. Boutaba. On balancing energy consumption in wireless sensor networks. *Vehicular Technology, IEEE Transactions on*, 58(6):2909–2924, 2009.

[BCDV09] C. Buratti, A. Conti, D. Dardari, and R. Verdone. An overview on wireless sensor networks technology and evolution. *Sensors*, 9(9):6869–6896, 2009.

[BCM+08] Stefano Basagni, Alessio Carosi, Emanuel Melachrinoudis, Chiara Petrioli, and Z. Maria Wang. Controlled sink mobility for prolonging wireless sensor networks lifetime. *Wirel. Netw.*, 14(6):831–858, December 2008.

[Bet02] C. Bettstetter. On the minimum node degree and connectivity of a

wireless multihop network. In *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pages 80–91. ACM, 2002.

[BFN01] L. Barriére, P. Fraigniaud, and L. Narayanan. Robust position-based routing in wireless ad hoc networks with unstable transmission ranges. In *Proceedings of the 5th international workshop on Discrete algorithms and methods for mobile computing and communications*, pages 19–27. ACM, 2001.

[BSLC04] J. Beaver, M.A. Sharaf, A. Labrinidis, and P.K. Chrysanthis. Location-aware routing for data aggregation in sensor networks. *Geosensor Networks*, pages 189–209, 2004.

[CABM05] D.S.J.D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. *Wireless Networks*, 11(4):419–434, 2005.

[Cas] The castalia wireless sensor network simulator.

[CL73] K. M. Chandy and T. Lo. The capacitated minimum spanning tree. *Networks*, 3:173–181, 1973.

[CTC10] T.S. Chen, H.W. Tsai, and C.P. Chu. Adjustable convergecast tree protocol for wireless sensor networks. *Computer Communications*, 33(5):559–570, 2010.

[CTL+09] Y.J. Chu, C.P. Tseng, K.C. Liao, Y.C. Wu, F.M. Lu, J.A. Jiang, Y.C. Wang, C.L. Tseng, E.C. Yang, and K.Y. Ho. The first order load-balanced algorithm with static fixing scheme for centralized wsn system in outdoor environmental monitoring. In *Sensors, 2009 IEEE*, pages 1810–1813. IEEE, 2009.

[CZYG10] G. Chatzimilioudis, D. Zeinalipour-Yazti, and D. Gunopulos. Minimum-hot-spot query trees for wireless sensor networks. In *Proceedings of the Ninth ACM International Workshop on Data Engineering for Wireless and Mobile Access*, pages 33–40. ACM, 2010.

[DDK09a] K. Daabaj, M. Dixon, and T. Koziniec. Experimental study of load balancing routing for improving lifetime in sensor networks. In

Wireless Communications, Networking and Mobile Computing, 2009. WiCom'09. 5th International Conference on, pages 1–4. IEEE, 2009.

[DDK09b] K. Daabaj, MW Dixon, and T. Koziniec. Avoiding routing holes in homogeneous wireless sensor networks. *Lecture Notes in Engineering and Computer Science*, 2178(1):356–361, 2009.

[DEA06] I. Demirkol, C. Ersoy, and F. Alagoz. Mac protocols for wireless sensor networks: a survey. *Communications Magazine, IEEE*, 44(4):115–121, 2006.

[DH03] H. Dai and R. Han. A node-centric load balancing algorithm for wireless sensor networks. In *Global Telecommunications Conference, 2003. GLOBECOM'03. IEEE*, volume 1, pages 548–552. IEEE, 2003.

[ENR06] C. Efthymiou, S. Nikoletseas, and J. Rolim. Energy balanced data propagation in wireless sensor networks. *Wireless Networks*, 12(6):691–707, 2006.

[GJ79] M.R. Gary and D.S. Johnson. Computers and intractability: A guide to the theory of np-completeness, 1979.

[GKW+02] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex behavior at scale: An experimental study of low-power wireless sensor networks. Technical report, Technical Report UCLA/CSD-TR 02, 2002.

[Gla] Glacsweb.

[GLW03] W. Guo, Z. Liu, and G. Wu. Poster abstract: an energy-balanced transmission scheme for sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 300–301. ACM, 2003.

[HCB02] W.B. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *Wireless Communications, IEEE Transactions on*, 1(4):660–670, 2002.

[HCWC09] C. Huang, R.H. Cheng, T.K. Wu, and S.R. Chen. Localized routing

protocols based on minimum balanced tree in wireless sensor networks. In *Mobile Ad-hoc and Sensor Networks, 2009. MSN'09. 5th International Conference on*, pages 503–510. IEEE, 2009.

[HHKV01] P.H. Hsiao, A. Hwang, HT Kung, and D. Vlah. Load-balancing routing for wireless access networks. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 986–995. IEEE, 2001.

[HSX+12] Renjie Huang, Wen-Zhan Song, Mingsen Xu, Nina Peterson, Behrooz Shirazi, and Richard LaHusen. Real-world sensor network for long-term volcano monitoring: Design and findings. *IEEE Trans. Parallel Distrib. Syst.*, 23(2):321–329, February 2012.

[HX10] Yonggang He and Tingrong Xu. The research of non-uniform node distribution in wireless sensor networks. In *Information Engineering and Electronic Commerce (IEEC), 2010 2nd International Symposium on*, pages 1–6, july 2010.

[IDT] IDTechEx. Wireless sensor networks 2011-2021. http://www.idtechex.com/research/reports/wireless-sensor-networks-2011-2021-000275.asp. Accessed Sep 6 2012.

[JCH84] R. Jain, D.M. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. *Technical Report, Digital Equipment Corporation*, DEC-TR-301, 1984.

[JOW+02] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In *ACM Sigplan Notices*, volume 37, pages 96–107. ACM, 2002.

[KEW02] L. Krishnamachari, D. Estrin, and S. Wicker. The impact of data aggregation in wireless sensor networks. In *Distributed Computing Systems Workshops, 2002. Proceedings. 22nd International Conference on*, pages 575–578. IEEE, 2002.

[KF12a] A. Kleerekoper and N. Filer. The relay area problem in wireless

sensor networks. In *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, pages 1–5. IEEE, 2012.

[KF12b] A. Kleerekoper and N. Filer. Revisiting blacklisting and justifying the unit disk graph model for energy-efficient position-based routing in wireless sensor networks. *Wireless Days*, 2012.

[KF12c] A. Kleerekoper and N. Filer. Trading latency for load balancing in many-to-one wireless networks. In *Wireless Telecommunications Symposium (WTS), 2012*, pages 1–9. IEEE, 2012.

[KK00] B. Karp and H.T. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254. ACM, 2000.

[KNS05] J. Kuruvila, A. Nayak, and I. Stojmenovic. Hop count optimal position-based packet routing algorithms for ad hoc wireless networks with a realistic physical layer. *Selected Areas in Communications, IEEE Journal on*, 23(6):1267–1275, 2005.

[KS78] L. Kleinrock and J. Silvester. Optimum transmission radii for packet radio networks or why six is a magic number. In *Proceedings of the IEEE National Telecommunications Conference*, volume 4, pages 1–4. Birimingham, Alabama, 1978.

[KWZ03] F. Kuhn, R. Wattenhofer, and A. Zollinger. Ad-hoc networks beyond unit disk graphs. In *Proceedings of the 2003 joint workshop on Foundations of mobile computing*, pages 69–78. ACM, 2003.

[LBB05] S. Lee, B. Bhattacharjee, and S. Banerjee. Efficient geographic routing in multihop wireless networks. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 230–241. ACM, 2005.

[LH05] Jun Luo and J.-P. Hubaux. Joint mobility and routing for lifetime elongation in wireless sensor networks. In *INFOCOM 2005. 24th*

Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, volume 3, pages 1735–1746 vol. 3, march 2005.

[LM05] J. Li and P. Mohapatra. An analytical model for the energy hole problem in many-to-one sensor networks. In *IEEE Vehicular Technology Conference*, volume 62, page 2721. IEEE; 1999, 2005.

[LM07] J. Li and P. Mohapatra. Analytical modeling and mitigation techniques for the energy hole problem in sensor networks. *Pervasive and Mobile Computing*, 3(3):233–254, 2007.

[LNA05] Jie Lian, Kshirasagar Naik, and Gordon B. Agnew. Data capacity improvement of wireless sensor networks using non-uniform sensor distribution. *International Journal of Distributed Sensor Networks*, 2:121–145, 2005.

[LNN06] Yunhuai Liu, Hoilun Ngan, and L.M. Ni. Power-aware node deployment in wireless sensor networks. In *Sensor Networks, Ubiquitous, and Trustworthy Computing, 2006. IEEE International Conference on*, volume 1, page 8 pp., june 2006.

[LR02] S. Lindsey and C.S. Raghavendra. Pegasis: Power-efficient gathering in sensor information systems. In *Aerospace Conference Proceedings, 2002. IEEE*, volume 3, pages 3–1125. IEEE, 2002.

[LXG05] Z. Liu, D. Xiu, and W. Guo. An energy-balanced model for data transmission in sensor networks. In *Vehicular Technology Conference, 2005. VTC-2005-Fall. 2005 IEEE 62nd*, volume 4, pages 2332–2336. IEEE, 2005.

[Mac09] M. Macedo. Are there so many sons per node in a wireless sensor network data aggregation tree? *Communications Letters, IEEE*, 13(4):245–247, 2009.

[Mar] MarketsandMarkets. Marketsandmarkets: Industrial wireless sensor networks (iwsn) market worth $3.795 billion by 2017. http://www.marketsandmarkets.com/PressReleases/wireless-sensor-network.asp. Accessed Sep 6 2012.

[MCP+02] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk,

and John Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, WSNA '02, pages 88–97, New York, NY, USA, 2002. ACM.

[MF] S. McCanne and S. Floyd. ns network simulator.

[MP08] K.M. Martin and M. Paterson. An application-oriented framework for wireless sensor network key establishment. *Electronic Notes in Theoretical Computer Science*, 192(2):31–41, 2008.

[MR04a] V. Mhatre and C. Rosenberg. Design guidelines for wireless sensor networks: communication, clustering and aggregation. *Ad Hoc Networks*, 2(1):45–63, 2004.

[MR04b] V. Mhatre and C. Rosenberg. Homogeneous vs heterogeneous clustered sensor networks: a comparative study. In *Communications, 2004 IEEE International Conference on*, volume 6, pages 3646–3651. IEEE, 2004.

[ns3] The network simulator - ns-3.

[OS06] S. Olariu and I. Stojmenovic. Design guidelines for maximizing lifetime and avoiding energy holes in sensor networks with uniform distribution and uniform reporting. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12, April 2006.

[PBC10] E. Park, D. Bae, and H. Choo. Energy efficient geographic routing for prolonging network lifetime in wireless sensor networks. In *Computational Science and Its Applications (ICCSA), 2010 International Conference on*, pages 285–288. IEEE, 2010.

[PCH04] M. Perillo, Zhao Cheng, and W. Heinzelman. On the problem of unbalanced load distribution in wireless sensor networks. In *Global Telecommunications Conference Workshops, 2004. GlobeCom Workshops 2004. IEEE*, pages 74–79, Nov.-3 Dec. 2004.

[PH08] D. Puccinelli and M. Haenggi. Arbutus: Network-layer load balancing for wireless sensor networks. In *Wireless Communications and*

Networking Conference, 2008. WCNC 2008. IEEE, pages 2063–2068. IEEE, 2008.

[PH09] D. Puccinelli and M. Haenggi. Lifetime benefits through load balancing in homogeneous sensor networks. In *Wireless Communications and Networking Conference, 2009. WCNC 2009. IEEE*, pages 1–6. IEEE, 2009.

[Res] BCC Research. Global markets and technologies for wireless sensors – focus on emea. http://www.bccresearch.com/report/emea-wireless-sensors-markets-ias042a.html. Accessed 6 Sep 2012.

[RM04] K. Romer and F. Mattern. The design space of wireless sensor networks. *Wireless Communications, IEEE*, 11(6):54–61, 2004.

[Sad05] B.M. Sadler. Fundamentals of energy-constrained sensor network systems. *Aerospace and Electronic Systems Magazine, IEEE*, 20(8):17–35, 2005.

[SCL+08] C. Song, J. Cao, M. Liu, Y. Zheng, H. Gong, and G. Chen. Mitigating energy holes based on transmission range adjustment in wireless sensor networks. In *Proceedings of the 5th International ICST Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, page 32. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[SFP] Sfpark. http://sfpark.org/. Accessed Sep 6 2012.

[Sie] Siega system. http://www.siegasystem.com/en/index.html. Accessed Sep 6 2012.

[SL01] I. Stojmenovic and X. Lin. Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, pages 1023–1032, 2001.

[SML+04] G. Simon, M. Maróti, Á. Lédeczi, G. Balogh, B. Kusy, A. Nádas, G. Pap, J. Sallai, and K. Frampton. Sensor network-based countersniper system. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 1–12. ACM, 2004.

[SMP99] K. Sohrabi, B. Manriquez, and G.J. Pottie. Near ground wideband channel measurement in 800-1000 mhz. In *Vehicular Technology Conference, 1999 IEEE 49th*, volume 1, pages 571–574. IEEE, 1999.

[SNK05] I. Stojmenovic, A. Nayak, and J. Kuruvila. Design guidelines for routing protocols in ad hoc and sensor networks with a realistic physical layer. *Communications Magazine, IEEE*, 43(3):101–106, 2005.

[SO05] Ivan Stojmenovic and Stephan Olariu. *Data-Centric Protocols for Wireless Sensor Networks*, pages 417–456. John Wiley & Sons, Inc., 2005.

[SR02] R.C. Shah and J.M. Rabaey. Energy aware routing for low energy ad hoc sensor networks. In *Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE*, volume 1, pages 350–355. Ieee, 2002.

[SZHK04] K. Seada, M. Zuniga, A. Helmy, and B. Krishnamachari. Energy-efficient forwarding strategies for geographic routing in lossy wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 108–121. ACM, 2004.

[TK84] H. Takagi and L. Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *Communications, IEEE Transactions on*, 32(3):246–257, 1984.

[TM09] I. Tellioglu and H.A. Mantar. A proportional load balancing for wireless sensor networks. In *Sensor Technologies and Applications, 2009. SENSORCOMM'09. Third International Conference on*, pages 514–519. IEEE, 2009.

[Vol] Volcanosri: 4d volcano tomography in a large-scale sensor network.

[WBMP05] Z.M. Wang, S. Basagni, E. Melachrinoudis, and C. Petrioli. Exploiting sink mobility for maximizing sensor networks lifetime. In *System Sciences, 2005. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference on*, page 287a, Jan. 2005.

[WCD08] Xiaobing Wu, Guihai Chen, and S.K. Das. Avoiding energy holes in wireless sensor networks with nonuniform node distribution. *Parallel*

and Distributed Systems, IEEE Transactions on, 19(5):710–720, May 2008.

[Wes12] P. Wessa. Free statistics software, office for research development and education, version 1.1.23-r7. *URL http://www. wessa. net*, 2012.

[WOW+03] A. Wadaa, S. Olariu, L. Wilson, K. Jones, and Q. Xu. On training a sensor network. In *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, pages 8–pp. IEEE, 2003.

[WTC03] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 14–27. ACM, 2003.

[WWQ+10] Yongcai Wang, Yuexuan Wang, Xiao Qi, Liwen Xu, Jinbiao Chen, and Guanyu Wang. L3sn: A level-based, large-scale, longevous sensor network system for agriculture information monitoring. 2010.

[YX10] YoungSang Yun and Ye Xia. Maximizing the lifetime of wireless sensor networks with mobile sink in delay-tolerant applications. *Mobile Computing, IEEE Transactions on*, 9(9):1308–1318, Sept. 2010.

[ZG03] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 1–13. ACM, 2003.

[ZHKS04] G. Zhou, T. He, S. Krishnamurthy, and J.A. Stankovic. Impact of radio irregularity on wireless sensor networks. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 125–138. ACM, 2004.

[ZK04] M. Zuniga and B. Krishnamachari. Analyzing the transitional region in low power wireless links. In *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, pages 517–526. IEEE, 2004.

[ZS09] H. Zhang and H. Shen. Balancing energy consumption to maximize network lifetime in data-gathering sensor networks. *Parallel and Distributed Systems, IEEE Transactions on*, 20(10):1526–1539, 2009.